P. MELUSSON

INITIATION

A LA MICRO-INFORMATIQUE

LE MICRO PROCESSEUR

POCHE *informatique* 14

Collection
POCHE-INFORMATIQUE

Initiation
à la micro-informatique

LE MICROPROCESSEUR

ISBN 2-85535-086-7

ISSN 0757-6730

Pierre MÉLUSSON

•

Initiation
à la micro-informatique

LE MICROPROCESSEUR

(6e édition revue et corrigée)

•

Diffusion :
EDITIONS TECHNIQUES ET SCIENTIFIQUES FRANÇAISES
2 à 12, rue de Bellevue - 75940 PARIS CEDEX 19

COLLECTION ETSF MICRO-SYSTEMES

1 - A. VILLARD et M. MIAUX, *Un microprocesseur pas à pas*
2 - A. VILLARD et M. MIAUX, *Systèmes à microprocesseur*
3 - P. GUEULLE, *Maîtrisez votre ZX 81*
4 - E. FLOEGEL, *Du Basic au Pascal*
5 - P. COURBIER, *Vous avez dit Basic ?*
6 - M. MARCHAND, *Vous avez dit micro ?*
7 - P. GUEULLE, *Pilotez votre ZX 81*
8 - M. JACQUELIN, *La micro-informatique et son A-B-C*
9 - M. OURY, *Maîtrisez les TO 7 et TO 7-70*
10 - P. GUEULLE, *Pilotez votre Oric, 1 + Atmos*
11 - P. JOUVELOT et D. LE CONTE DES FLORIS, *Système d'exploitation et logiciel de base des micro-ordinateurs*
12 - P. GUEULLE, *Robotisez votre ZX 81*

COLLECTION POCHE *Informatique*

1 - G. ISABEL, *50 programmes pour ZX 81*
2 - P. GUEULLE, *Montages périphériques pour ZX 81*
3 - C. GALAIS, *Passeport pour Applesoft*
4 - R. BUSCH, *Passeport pour Basic*
5 - M. ROUSSELET, *Mathématiques sur ZX 81*
6 - C. GALAIS, *Passeport pour ZX 81*
7 - G. PROBST, *50 programmes pour Casio FX-702 et 801 P*
8 - G. PROBST, *60 programmes pour Casio PB-100*
9 - M. SAAL, *Utilitaires pour ZX 81*
10 - C. GALAIS, *Passeport pour Commodore 64*
11 - D. RANC, *Assembleur du TRS 80*
12 - D. LASSERAN, *30 programmes pour Commodore 64*

Table des matières

CHAPITRE 1 - **Introduction** .... **9 à 11**

Du calculateur à l'ordinateur — De l'ordinateur au micro-ordinateur .... 9 à 11

CHAPITRE 2 - **Le cerveau humain et l'ordinateur : cerveau robot** .... **12 à 19**

Le Hardware et le Software 12
Le premier centre : la mémoire .... 12 et 13
Le deuxième centre : le raisonnement .... 13 à 19
Différences entre l'homme et la machine 19

CHAPITRE 3 - **Ordinateur — Calculateur — Microprocesseur — Qui êtes-vous?** .... **20 à 26**

Les divers types d'ordinateurs : analogiques — digitaux — hybrides .... 20 et 21
Les ordinateurs et calculateurs digitaux .... 21 à 24
Le microprocesseur et son choix .... 24 à 26

CHAPITRE 4 - **Les langages d'ordinateurs — Les appareils de décodage — Des langages en « binaire pur »** .... **27 à 30**

Tableau des principaux langages de programmation 28

CHAPITRE 5 - **Le calcul binaire — Les codages : Octal, Hexadécimal et BCD** .... **31 à 39**

Tableau des correspondances entre le système décimal et les systèmes octal, hexadécimal et binaire naturel 32

Tableau des correspondances entre le système décimal et les systèmes binaire naturel et BCD 33
Conversion décimal/binaire et binaire/décimal ...... 34 à 35
Les 4 opérations en système binaire : L'addition, la soustraction, la multiplication et la division .... 35 à 37
Conversion : programme-source/programme-objet . 37 à 39

CHAPITRE 6 - **Les fonctions logiques** .... **40 à 57**

Les fonctions logiques combinatoires .... 43 à 50
Les fonctions logiques séquentielles .... 50 à 57

CHAPITRE 7 - **La technologie des microprocesseurs** ..... **58 à 70**

Étude des diverses technologies (configurations générales des circuits et procédés de fabrication) .... 58 à 68
Tableau des microprocesseurs .... 69 et 70

CHAPITRE 8 - **Organisation des microprocesseurs** .... **71 à 99**

Les diverses parties composantes d'un microprocesseur .... 71 à 79
Étude du microprocesseur MC6800 .... 79 à 99

CHAPITRE 9 - **Évolution technique des microprocesseurs et des micro-ordinateurs** .... **100 à 113**

Introduction .... 100 et 101
Vers l'intégration totale des circuits .... 101 à 103
Vers la réalisation très économique de problèmes simples .... 104 et 105
Vers le développement des circuits très sophistiqués 104 à 113
Les microprocesseurs montés en tranches de 4 bits 113

CHAPITRE 10 - **Les mémoires** .... **114 à 126**

Les divers types de mémoires .... 114 et 115
1. Caractéristiques et définitions des paramètres électriques des mémoires à C.I. .... 115 à 119
2. Les diverses catégories et leur fabrication de mémoires à C.I. .... 119 à 123
3. Étude des mémoires faisant partie de la famille Motorola-M6800 .... 123 à 126

CHAPITRE 11 - **Les circuits et systèmes d'interface entre microprocesseur/mémoires unité centrale et périphériques** **127 à 146**

Utilité des systèmes d'interface .... 127 et 128
Les diverses possibilités de transfert .... 129 et 130
Exemples de circuits d'interface .... 131 à 139
Vers des périphériques économiques .... 139 et 140
Les moniteurs TV M68MDM1 et M68MDM9 .... 141 à 144
Le générateur de caractères vidéo - Motorola - MC 6847 .... 144 à 146

CHAPITRE 12 - **La programmation — Comment bâtir un micro-ordinateur en vue d'une utilisation donnée** .... **147 à 158**

La conception d'un programme .... 147 à 152
Résolution d'un problème pratique simple .... 152 à 154
Les outils de travail .... 155 à 158

CONCLUSION 159
BIBLIOGRAPHIE 160

CHAPITRE **1** *Introduction*

Du calculateur à l'ordinateur De l'ordinateur au micro-ordinateur

La science informatique est actuellement, dans ses applications essentielles, le domaine des ***Calculateurs*** et des ***Ordinateurs.***

Un rapide coup d'œil sur l'évolution du passé jusqu'à nous de cette science permettra de mieux situer son contexte de développement. Il est évidemment lié aux techniques de réalisations utilisées, soit dans un ordre de chronologie historique :

1. Le temps de la mécanique pure

C'est antérieurement à l'ère chrétienne, soit environ 500 ans avant Jésus-Christ que les Chinois inventent le premier instrument à calculer : le ***boulier.*** Il leur permet de compter rapidement et avec précision. Le boulier est encore employé de nos jours en certains endroits de notre planète. Il sert de support matériel à l'homme afin d'effectuer un calcul. Pour s'en servir judicieusement il doit déployer et sa force musculaire et son intelligence.

Il n'en est plus de même lorsqu'en 1642 ***Blaise Pascal invente une première calculatrice*** qui comporte son mécanisme propre de fonctionnement.

Dans le même siècle ***Jacquard*** invente le ***métier à tisser programmé à l'aide de cartes perforées.*** Cette idée de programmation prélude déjà au fonctionnement de certains chargeurs périphériques actuels d'ordinateurs.

2. Le temps de l'électro-mécanisme

En 1924 l'ingénieur Norvégien ***Bull*** prend un brevet concernant un ensemble électro-mécanique et c'est en 1936 que les Américains utilisent un calculateur dont l'automatisme est assuré grâce à des ***relais électro-mécaniques.***

3. Le temps de l'électronique

Ce temps peut, du fait des composants actifs successivement employés, être divisé en trois phases successives :

a) **LE TEMPS DES TUBES ELECTRONIQUES.** Il verra en 1950 la naissance de l'***Univac 1.*** C'est le premier ordinateur commercial. Il se distingue du calculateur par un jeu d'instructions assurant l'établissement de nombreux programmes. L'élaboration de programmes permet en effet à la machine ordinateur de résoudre les problèmes diversifiés demandés.

b) **LE TEMPS DES SEMI-CONDUCTEURS.** (diodes et transistors). Il se situe aux alentours de 1960, date à laquelle apparaît sur le marché les premiers ordinateurs entièrement transistorisés. Ils comportent généralement de 5 000 à 10 000 transistors et autant de diodes.

c) **LE TEMPS DES CIRCUITS INTEGRES**

Ce sont plus particulièrement les circuits appelés ***LSI*** c'est-à-dire : « Large Scale Integration ». Ils peuvent contenir chacun de 15 000 à 20 000 jonctions diffusées sur une seule petite pastille de silicium de quelques dizaines de mm^2 de surface.

Grâce à ces circuits on va ainsi pouvoir passer du domaine de l'informatique à celui de la micro-informatique. C'est d'abord l'ère de la ***calculatrice de poche,*** puis celle du ***micro-ordinateur.***

L'âme de ce micro-ordinateur est le ***microprocesseur.*** Il est l'organe chargé de résoudre les opérations arithmétiques et les fonctions logiques suivant les informations qu'il traite d'une ***manière séquentielle.***

En 1971 apparaît le premier microprocesseur sur le marché. En 1973 c'est au tour du premier micro-ordinateur équipé d'un microprocesseur : le ***Micral*** de la société française ***R2E.***

L'ordinateur est un gros engin dont le prix de revient est très élevé. Il est par contre dans ses possibilités de résoudre facilement en quantités et en qualités les problèmes les plus divers, aussi, afin d'être rentable, il doit fonctionner régulièrement à pleine charge et éviter de compter des temps morts. Pour ces raisons, il ne peut être utilisé que par des sociétés commerciales, industrielles ou financières très importantes.

Dans les sociétés moins importantes, des terminaux peuvent être aménagés. Ils correspondent alors avec l'ordinateur central auquel ils sont rattachés à l'aide de lignes spécialement aménagées que les sociétés louent au propriétaire de l'ordinateur central. On indique dans ce cas que les terminaux travaillent en ***temps partagé,*** ce qui se dit souvent : en ***Time Sharing.***

Les ***terminaux*** prennent chacun, suivant un ordre de priorité établi en fonction de l'urgence et de la qualité de leurs demandes, une partie du temps propre à l'ordinateur central.

Cependant, là encore, il faut bien admettre qu'un tel système d'exploitation exige dans son ensemble une organisation massive et coûteuse.

Le micro-ordinateur est en revanche, comparativement à l'ordinateur, d'un prix de revient nettement meilleur marché et ceci grâce, en grande partie, à *la souplesse d'organisation de ses structures,* pouvant s'adapter parfaitement, après une étude sérieuse de composition minimum, aux seuls besoins spécifiques désirés.

Sa taille miniaturisée et sa faculté d'adaptation économiquement très avantageuses, relevant de la simplicité de ses circuits, doivent permettre dans les années à venir, un développement sans précédent dans tous les domaines de l'électronique, aussi bien, dans ceux de la ***logique*** que dans ceux professionnels des ***télécommunications,*** des ***automatismes industriels,*** de la ***télévision,*** de l'***électroménager*** et dans bien d'autres secteurs encore qu'il nous sera donné dans le temps et par la suite de découvrir.

CHAPITRE 2

Le cerveau humain et l'ordinateur : cerveau robot

L'ordinateur est un cerveau robot présentant des analogies avec le cerveau humain.

Afin de les définir, il faut donc imaginer ce qu'est le cerveau humain et établir les parallèles, avec le cerveau robot d'un ordinateur.

C'est ce que maintenant, nous allons tenter d'effectuer.

Sur le plan matériel, c'est un organe. C'est la partie ***hardware*** du cerveau, autrement dit : la partie ***concrète*** et palpable du cerveau.

Comment, dans cet organe, ce « hardware », s'élabore le cheminement des idées, le raisonnement qui conduit à commander notre corps... c'est ce qu'il faut essayer de comprendre... !

Toute cette élaboration des idées constitue le ***software*** du cerveau, c'est-à-dire sa partie ***abstraite,*** non palpable.

Ainsi pour l'***informaticien*** :

Le concret : c'est-à-dire le circuit, tout ce que l'on peut voir, toucher, c'est le **hardware.**

L'abstrait : c'est-à-dire la partie immatérielle de la science, la matière grise, c'est le **software.**

La pensée humaine, pour se forger et s'exprimer a besoin de deux centres constitutifs distincts :

Le premier centre est celui de la ***mémoire.*** C'est le lieu où toutes les informations acquises sont classées dans des « *petites cases* » parfaitement bien ordonnées.

La capacité totale de la mémoire est « *variable* » mais non « *infinie* ».

De plus, certaines informations, ou ***données,*** ou bien encore ***data*** doivent être stockées une fois pour toutes et ne pas s'effacer. Ces « *data* » seront soigneusement entassées dans une mémoire ***ROM*** (Read Only Memory). C'est une mémoire à ***lecture*** seulement ***indestructible.***

Cette mémoire figée une fois pour toutes est exprimée en français par le terme ***mémoire morte.***

Par contre, d'autres informations, qui ne sont pas des données premières, mais plutôt le résultat de la composition ordonnée de plusieurs de celles-ci, n'ont pas besoin d'être éternellement à demeure ; on peut très bien n'avoir à s'en servir qu'une fois. On doit alors pouvoir :

1. S'en servir.

2. Les effacer de manière à laisser des cases vides nécessaires à l'accumulation de nouvelles informations.

Toutes ces informations fugitives ou temporaires seront ***écrites*** puis ***effacées*** dans des ***RAM*** (Random Access Memory).

En Français, on les appelle, par opposition avec les mémoires mortes, des ***mémoires vives.***

Ce sont dans le cerveau robot, des mémoires d'où l'on peut extraire les données pour les lire, puis les effacer en vue de laisser de la place libre qui sera ensuite utilisée afin de stocker de nouvelles informations qui y seront à nouveau écrites.

La lecture d'une donnée extraite ou l'écriture d'une donnée introduite s'opère grâce à la fonction ***Read/Write*** de l'ordinateur.

Le deuxième centre de notre cerveau est celui du ***raisonnement.*** C'est celui dans lequel s'élabore le cheminement des idées. Ce centre est capable d'effectuer soit :

— Un raisonnement ***logique*** (il résout les problèmes concernant les ***fonctions logiques***) ou soit encore :

— Un raisonnement de calcul que les informaticiens appellent un ***algorithme*** (un procédé de calcul). (Il résout les problèmes concernant les ***opérations arithmétiques***).

Ce centre est appelé dans le micro-ordinateur, c'est-à-dire dans notre cerveau robot : le ***microprocesseur.***

Les abréviations pour le nommer sont :

Le ***CPU*** (Central Processing Unit) c'est-à-dire : l'unité centrale de traitement de l'ordinateur, ou bien encore le ***MPU*** (Microprocessing Unit) c'est-à-dire : unité de traitement du microprocesseur.

Le centre à l'intérieur même du microprocesseur, où s'exécute véritablement la résolution des problèmes logiques et arithmétiques se désigne par le terme ***ALU*** (Arithmetic and Logic Unit).

Pour effectuer un raisonnement, par exemple l'opération arithmétique (4 + 5), le centre de raisonnement (le microprocesseur de notre cerveau

robot) doit aller chercher dans les mémoires où elles sont stockées, les informations 4, puis 5 et enfin l'***instruction*** (+). Pour cela, il faut que des fibres conductrices existent entre le MPU et les endroits des mémoires où sont stockées les différentes « data ». Ces fibres s'appellent des ***BUS.*** Il existe deux sortes principales de bus ou encore en français de ***faisceaux de lignes.***

1. L'***Adress Bus*** et 2. La ***Data Bus.***

L'Adress Bus (le bus d'adresses) est le faisceau de lignes qui relie le MPU aux adresses où sont contenues dans la mémoire les data recherchées.

L'Adress Bus est généralement ***unilatéral,*** c'est-à-dire que le cheminement dans ses lignes ne s'exerce que dans le sens :

MPU → MEMOIRE

La Data Bus (le bus d'informations) est généralement : ***bilatéral*** c'est-à-dire qu'elle peut s'exercer soit dans le sens : **MEMOIRE → MPU** quand l'information a été recherchée par le MPU à son adresse réelle, ou bien encore dans le sens : ***MPU → MEMOIRE*** lorsque le résultat d'une opération vient d'être effectuée dans l'unité centrale de traitement et a besoin d'être stockée à une adresse libre d'une mémoire « RAM ».

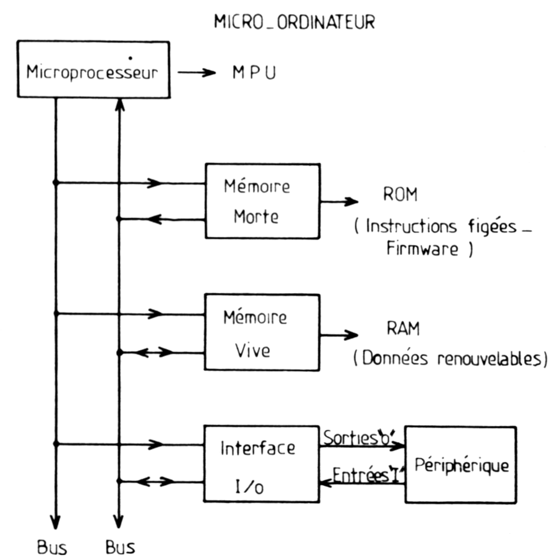


Figure 1

Dans notre opération : 4 et 5 sont les « data » ou données arithmétiques. L'« algorithme » ou *« procédé de calcul »* est défini par les signes (+) et (=). Ces signes constituent « l'instruction » du calcul à effectuer. L'organisation du travail de l'ordinateur est le fait du ***logiciel.***

Cette organisation procède par « instructions ». L'ensemble des instructions d'un micro-ordinateur s'appelle un ***jeu d'instructions.***

L'ensemble des instructions nécessaires à résoudre une fonction s'appelle un ***programme.***

Certains programmes sont complexes. Afin de les simplifier, on peut les diviser en ***sub-routines*** ou ***sous-programmes*** généralement « tout fait ». C'est de l'assemblage de méccano...! Celui qui réalise un programme est appelé un ***programmateur*** ou encore quelquefois d'une façon plus abréviative un ***programmeur.*** Toutes les opérations d'un programme se déroulent séquentiellement suivant un ordre bien établi.

Elles sont synchronisées dans le temps et contrôlées par des ***signaux d'horloge*** ou ***clock*** qui peuvent être obtenus à partir d'un oscillateur à « quartz ».

Les organes de commande ou de contrôles d'opérations peuvent être ***activés*** par des signaux amenés à ceux-ci par l'intermédiaire d'un « bus » appelé ***bus de commande*** ou ***bus de contrôle.***

Toute cette organisation d'un micro-ordinateur a pour support le ***système binaire*** qui se prête bien à un signal électrique travaillant par ***tout ou rien.***

Les « fonctions logiques » comme les « calculs » sont exprimés dans ce système, ne comportant seulement que les chiffres 0 et 1 pour désigner tout le contenu d'un programme donné.

L'information élémentaire du système binaire s'appelle le ***bit*** (par contraction du mot « Binary Digit »).

Un mot est composé d'un certain nombre de « bits ». Plus il peut comporter de « bits » et plus il est précis.

Dans l'inscription d'un mot dans une mémoire, le bit de droite est le : ***LSB*** (Less Significant Bit) c'est-à-dire le bit le moins significatif, encore appelé : ***Bit de moindre poids.***

Le bit le plus à gauche est le : ***MSB*** (Most Significant Bit) c'est-à-dire le bit le plus significatif encore appelé : ***bit de plus fort poids.***

Les microprocesseurs comportent généralement des mots de 4 - 8 et 16 bits. La plupart actuellement sont structurés avec des mots de 8 bits.

Un mot de 8 bits est appelé un ***octet*** ou bien encore un ***byte.***

Enfin tout cerveau robot ne peut pas vivre uniquement en circuit fermé, tout d'ailleurs comme le cerveau d'un humain.

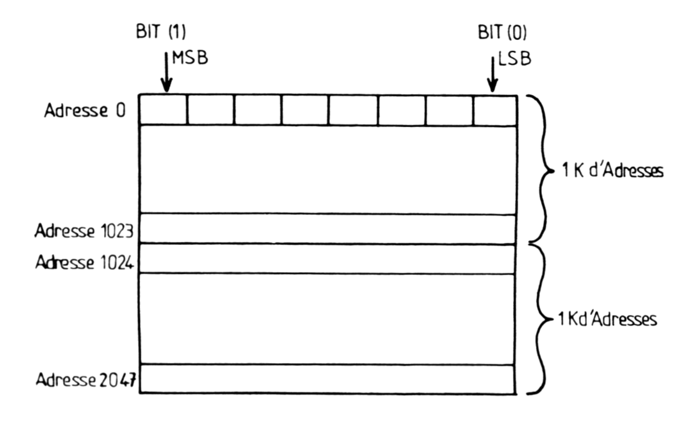


Figure 2 Représentation d'une mémoire de 2K mots de 8 bits

Ce dernier, en effet, a souvent besoin d'échanger ses idées avec ses semblables au cours d'une conversation. Par rapport à nous-mêmes notre voisin extérieur à notre « MOI » constitue un ***périphérique.*** Lorsqu'il nous pose une question, elle nous parvient suite à l'expression de sa propre pensée. Celle-ci ne peut pas être en tous points identique à la nôtre et afin de la comprendre et relier son « logiciel » au nôtre, nous avons besoin d'un ***interface*** entre notre centre de raisonnement c'est-à-dire notre CPU et celui du périphérique qui adapte notre système de raisonnement à celui de notre interlocuteur.

Cet interface peut avoir plusieurs noms :

C'est le ***I/O*** (input/output) de :

Input → Entrée des « data » provenant d'un périphérique.

Output → Sortie des « data » vers un périphérique.

C'est encore le ***PIA*** (Péripheral Interface Adapter).

Nous recevons généralement de la part d'un périphérique soit une demande de « data », soit une demande de « calcul » ou de « logique ».

S'il s'agit d'une simple demande de « data », notre cerveau robot peut avoir la possibilité de mettre en position d'entrée : ***THREE STATE CONTROL (TSC)*** le CPU, c'est-à-dire de mettre son entrée en condition d'impédance infinie. Par contre le PIA se branche directement sur la mémoire du micro-ordinateur comprenant la data demandée. On dit alors que le PIA travaille en ***DMA*** (Direct Memory Access) c'est-à-dire en accès direct à la mémoire.

S'il s'agit d'une demande de « calcul » ou de « logique », le PIA se branche directement à notre MPU.

Au moment où notre « MPU » perçoit une demande en provenance du périphérique via le PIA et qu'il est déjà en cours d'exécution d'un programme, 2 solutions peuvent alors être envisagées :

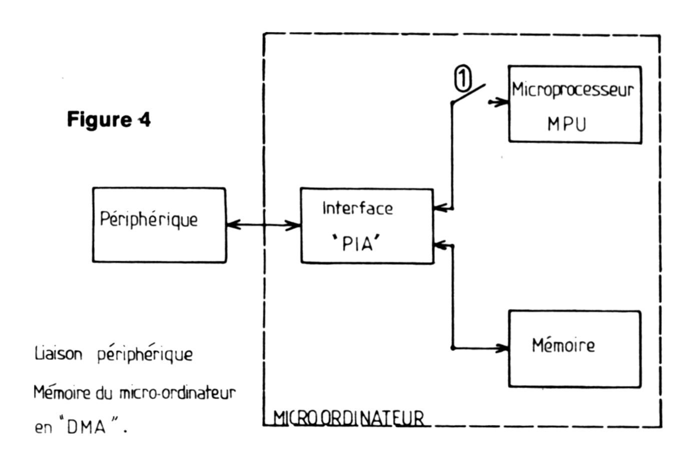
Solution 1. Le problème en cours de résolution est plus important que la demande extérieure. Il continue donc de se dérouler normalement jusqu'à épuisement et la demande extérieure est mise en attente pendant ce temps.

Solution 2. La demande provenant de l'extérieur est plus pressée à répondre que le programme qui se déroule actuellement dans le « MPU ».

En ce cas, il se produit un ***interrupt*** (une interruption). Le programme actuel s'arrête. Ce qui a déjà été réalisé peut-être mis en mémoire afin d'être conservé dans une mémoire spéciale appelée ***LIFO*** (last input, first output) ; c'est-à-dire : l'information dernière rentrée est aussi la première ressortie. La demande extérieure est alors programmée, puis l'ancien programme est ensuite repris à l'endroit où il en était resté et poursuivi jusqu'à sa conclusion finale.

Et puisque nous avons comparé l'homme et la machine, résumons en quelques mots, cette analogie ;

Figure 4



Liaison périphérique
Mémoire du micro-ordinateur
en "DMA".

(1) → en TSC - (Impédance infinie)

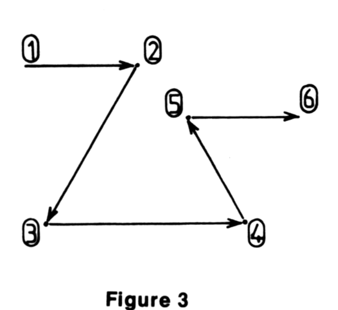


Figure 3

(1) → programme en cours de traitement.
(2) → Intervention d'un phénomène extérieur exigeant une interruption.
(3) → le programme est interrompu.
(4) → L'interruption est traitée
(5) → On revient au programme (1)
(6) → le programme (1) est repris et son traitement achevé.

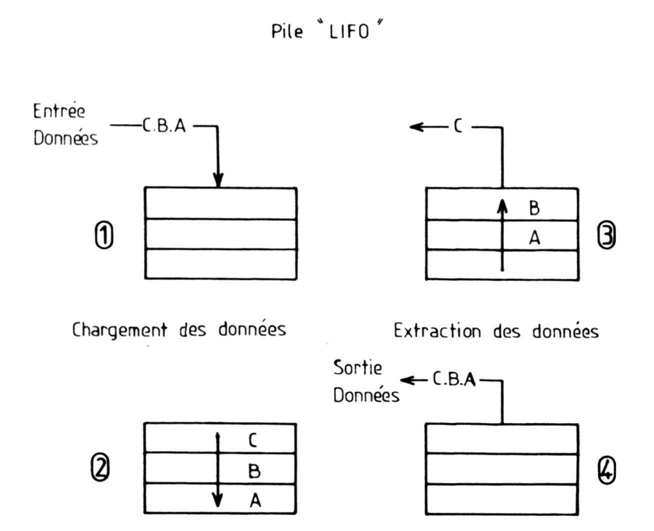


Fig. 5. — *La dernière information introduite est la première extraite, car On empile les « DATA » arbitrairement par le haut et on les vide également par le haut.*

On dit qu'une mémoire « LIFO » est une ***pile*** *ou bien encore un* ***registre à empilement.***

Soit à réaliser l'addition de plusieurs nombres :

L'organe périphérique du micro-ordinateur dans ce système peut être assimilé à la feuille de papier sur laquelle est inscrite la liste des nombres à additionner.

La feuille de papier est en effet ***la source*** des données à traiter. Lorsque le résultat s'inscrira dessus, elle sera aussi le lieu de ***destination.***

Les yeux qui voient les données à traiter et la main qui inscrit le résultat constituent le ***système d'interface (I/O).***

Les yeux rentrent les nombres en ***mémoire.***

Les mémoires, que ce soit celle de l'homme ou celle de l'ordinateur jouent le même rôle. Elles contiennent toutes deux l'***algorithme*** de l'addition.

La portion du cerveau qui réalise l'addition est son centre de raisonnement, le ***CPU*** de l'ordinateur réalise la même fonction.

Et pourtant, il existe entre l'homme et la machine deux différences essentielles :

1. La machine a la possibilité de **résoudre sans peine et à de très grandes vitesses des tâches répétitives** sans se fatiguer, ce que l'homme ne peut pas faire.

2. **L'homme a l'idée de création** grâce à son intuition et à son imagination. La machine en est totalement dépourvue, ce qui oblige l'homme à lui dicter un programme dans ses moindres détails. Ainsi la machine est encore reléguée à un niveau très inférieur à celui de l'homme.

Afin de mieux comprendre ce texte, nous l'avons illustré par les schémas synoptiques suivants :

Figure 1. Composition synoptique d'un micro-ordinateur.

Figure 2. Représentation synoptique d'occupation d'une mémoire.

Figure 3. Vecteurs de fonctionnement d'une demande d'interruption de programme.

Figure 4. Le PIA travaille en DMA à la demande d'un périphérique.

Figure 5. Fonctionnement d'une mémoire pile « LIFO ».

CHAPITRE **3**

Ordinateur Calculateur - Microprocesseur Qui êtes-vous ?

Un ordinateur est une machine capable de prendre des décisions et de réaliser des opérations arithmétiques ou logiques disponibles à sa sortie et conformes aux demandes programmées présentées à son entrée.

On peut distinguer 3 sortes de types différents d'ordinateurs :

1. Les ordinateurs analogiques

Dans ces ordinateurs, l'information se présente et est traitée ***sous forme analogique.***

Exemple d'une forme analogique :

Soit à résoudre la fonction : $Y = ax + b$.

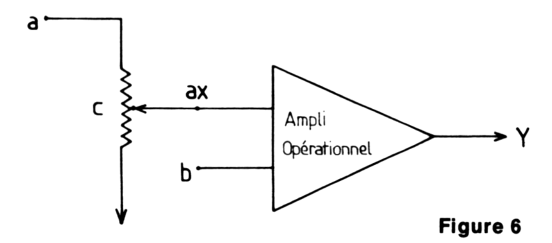


Figure 6

Le schéma de principe (Fig. 6) montre le circuit (hardware) d'ordinateur permettant de résoudre la fonction en ***une seule séquence.*** (x) est la ***variable*** et l'amplitude électrique de son signal est proportionnelle à la détermination de la position de la prise potentiométrique (point c).

On peut, en conclusion, affirmer qu'il y a proportionalité entre la valeur calculée de la fonction (Y) et celle de l'amplitude du signal électrique qui lui correspond.

2. Les ordinateurs digitaux.

Dans ces ordinateurs, l'information se présente et est traitée ***sous forme digitale*** (suite de niveaux électriques *B* BAS et *H* HAUT définissant les nombres binaires 0 et 1).

Exemples de traitement sous forme binaire : ceux-ci vont être donnés dans les paragraphes qui suivent.

3. Les ordinateurs hybrides.

Dans ces ordinateurs les 2 types d'informations électriques (analogique et digitale) sont combinées.

4. Ordinateurs et calculateurs digitaux :

Un ordinateur comme un calculateur, de type « digital » est capable sous forme binaire de stocker et de traiter des informations.

Il appartient donc alors de définir plus clairement la différence existant entre un calculateur et un ordinateur.

Un calculateur remplit seulement la fonction qui lui a été définie, de façon répétitive, lors de son câblage. C'est un système Hardware.

Son câblage, associant un certain nombre de circuits digitaux élémentaires : ***portes*** et ***flip-flop,*** constitue ce que l'on a coutume d'appeler : ***une logique câblée.***

La logique câblée se réalise généralement sur un circuit imprimé désigné sous le nom de ***carte.***

Exemple de logique câblée : soit à résoudre dans un calculateur digital la fonction : $\boldsymbol{Y = A.B + C}$.

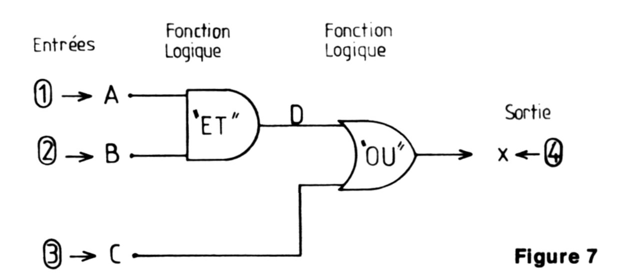


Figure 7

Le schéma de principe, figure 7, montre le circuit de logique câblée permettant de résoudre cette fonction en ***une seule séquence.***

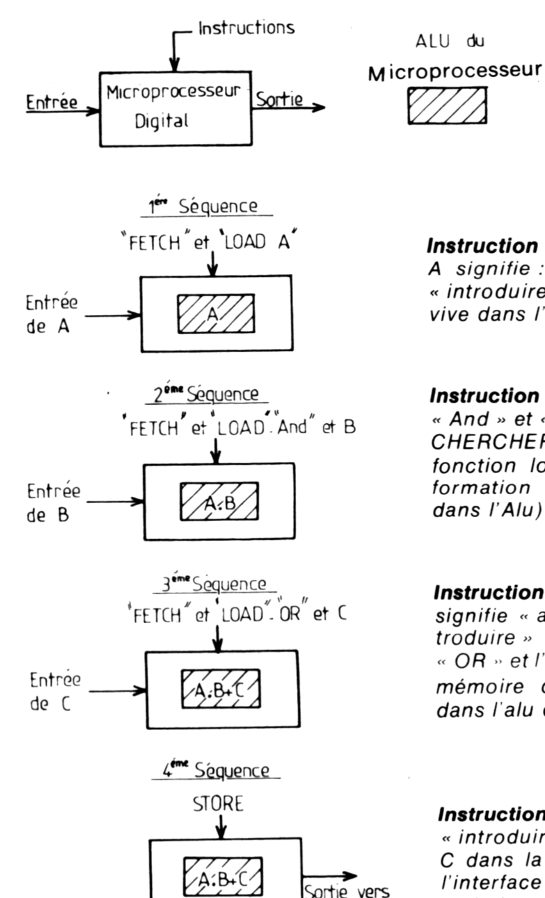


Figure 8

Instruction « FETCH » *et* « LOAD » *A signifie : « aller chercher » et « introduire » A (de la mémoire vive dans l'alu).*

Instruction « FETCH » *et* « LOAD » *« And » et « B » signifie : « ALLER CHERCHER » et « introduire » la fonction logique « And » et l'information « B » (de la mémoire dans l'Alu)*

Instruction « FETCH » et « LOAD » *signifie « aller chercher » et « introduire » la fonction logique « OR » et l'information « C » (de la mémoire dans l'alu). Formation dans l'alu de A.B + C.*

Instruction « STORE » *signifie « introduire » le résultat X = A.B + C dans la mémoire vive ou vers l'interface I/O (de l'alu dans la mémoire)*

Si l'on présente sur les entrées ① ② et ③ suivant un ***mode parallèle,*** c'est-à-dire ***simultanément,*** les informations binaires A, B et C, on réalisera sur la sortie ④ la valeur ***y*** recherchée.

Un ordinateur a la faculté de remplir n'importe quelle tâche dictée par une suite d'instructions. (Système software).

C'est sa différence essentielle avec un calculateur.

En contrepartie, son utilisation est plus compliquée car pour ainsi programmer une suite d'instructions, il devient auparavant nécessaire de connaître le langage assimilé par l'ordinateur.

Exemple de séquences d'instructions programmées :

Reprenons notre fonction ***X* = *A.B* + *C*** et résolvons-là maintenant à l'aide d'un ordinateur.

Le schéma synoptique des séquences d'instructions de la figuré 8 nous montre comment est résolu ce problème.

On pourrait résumer d'une façon très simple par le schéma de la figure 9 la différence essentielle de fonctionnement entre une « logique câblée » et un ordinateur.

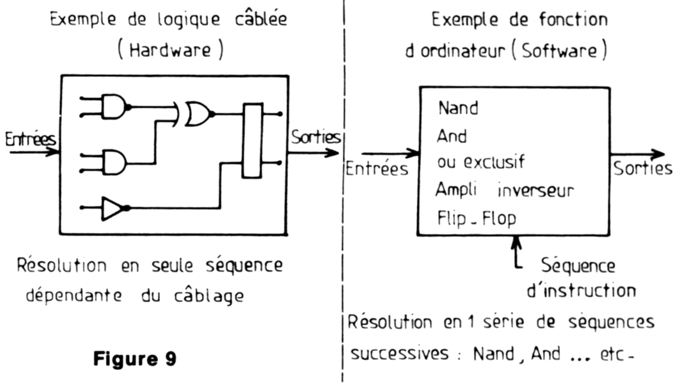


Figure 9

On trouve actuellement, dans le commerce, par classifications d'applications envisagées, trois types d'ordinateurs digitaux :

— les gros ordinateurs

— les mini-ordinateurs

— les micro-ordinateurs

Ils ont tous la même structure principale, mais ce qui, essentiellement les diffère entre eux est leur taille c'est-à-dire :

— ***leur capacité de mémoire***

— ***la grandeur de leur mot*** (nombre de bits d'un mot)

— ***le nombre et la qualité*** de leurs instructions disponibles

— ***la vitesse d'exécution*** des instructions.

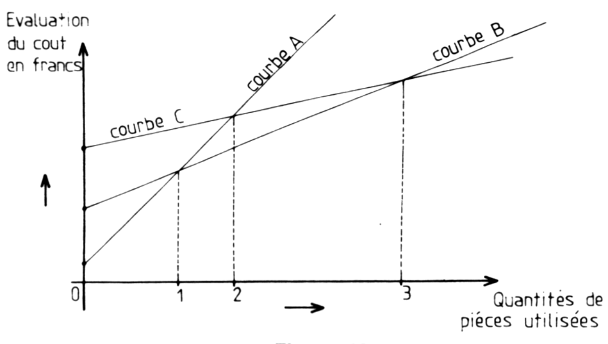


Figure 10

Nous distinguons en abscisses :
Dans la zone 0,1 - Une préférence pour la logique câblée (A)
Dans la zone 1,3 - Une préférence pour le microprocesseur (B)
Dans la zone supérieure à 3 - Une préférence pour le Custom Design (C)
A noter que dans la zone 1,2, si le microprocesseur est conseillé, on peut toutefois, à défaut, opter pour une logique câblée, meilleur marché en prix de revient que le custom design.

Ces performances diverses font que ces trois catégories principales d'ordinateurs diffèrent essentiellement dans leur prix et par le champ d'applications qu'ils peuvent couvrir.

Nous allons surtout considérer dans notre exposé les micro-ordinateurs digitaux qui sont ici l'objet de notre cours d'initiation.

A tout seigneur, tout honneur... parlons du microprocesseur, l'âme du micro-ordinateur.

Il est important maintenant d'en donner une définition plus précise que celle que nous en avons déjà faite.

L'union technique de l'électricité *(UTE)* a proposé au nom de la France au *C.E.I.* (Comité Électrotechnique International) la définition très complète suivante :

« Un microprocesseur à circuit intégré est un circuit intégré numérique capable après identification d'une séquence d'instructions, d'effectuer des opérations de traitement de l'information. »

Un microprocesseur :

1. Accepte des informations de données codées pour traitement et/ou stockage.

2. Effectue l'ensemble des opérations logiques, arithmétiques et de mouvement sur les informations d'entrée et/ou sur toutes les informations stockées.

3. Restitue en sortie les informations codées résultant du traitement effectué.

4. Peut recevoir et/ou fournir des informations relatives à son fonctionnement ou à son état.

Pour terminer ce chapitre, établissons dans un graphique une comparaison intéressante du coût entre

Courbe A. Une logique câblée avec des éléments standard ***(opérateurs, bascules, registres simples,*** etc.)

Courbe B. Un microprocesseur.

Courbe C. Une logique particulière, étudiée spécialement pour un client et pour ses besoins spécifiques. Il s'agit d'un ***Custom Design*** (Désignation anglaise du circuit particulier).

Cette comparaison en figure 10 du prix de revient est établie en fonction du nombre de pièces à négocier.

Tout ceci nous amène à penser qu'avant de prendre une décision sur l'emploi d'un de ces 3 matériels, une étude de rentabilité préalable s'avère nécessaire.

Le synoptique suivant présenté (Fig. 11) va maintenant nous permettre de réaliser en fonction des critères électriques de notre utilisation, le choix judicieux du matériel à adopter entre une « logique câblée » ou un « microprocesseur ».

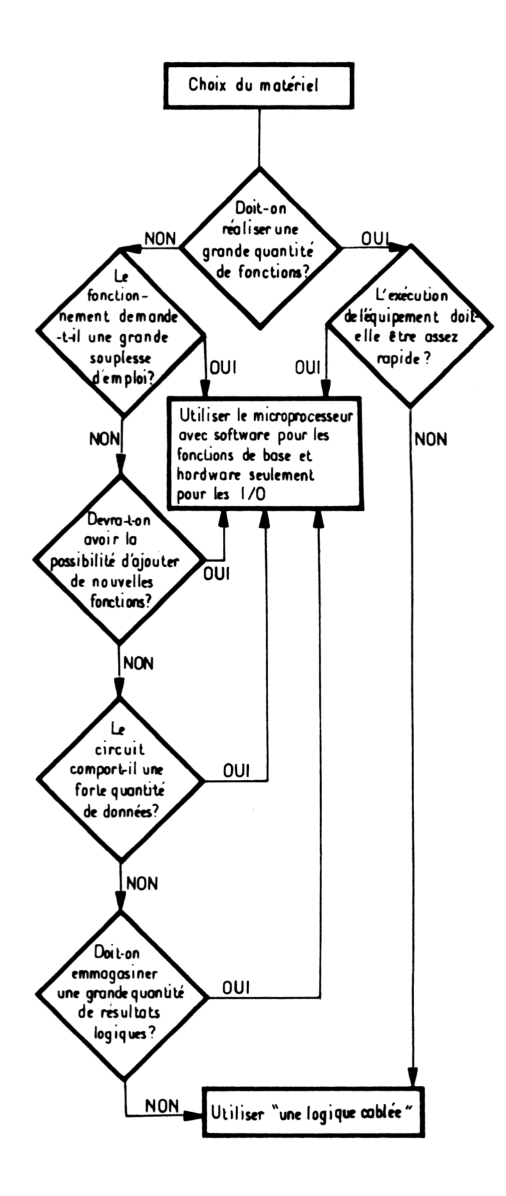

Choix du matériel
Doit-on réaliser une grande quantité de fonctions?
NON
OUI
Le fonctionnement demande-t-il une grande souplesse d'emploi?
L'exécution de l'équipement doit-elle être assez rapide?
OUI
OUI
Utiliser le microprocesseur avec software pour les fonctions de base et hordware seulement pour les I/O
NON
NON
Devra-t-on avoir la possibilité d'ajouter de nouvelles fonctions?
OUI
NON
Le circuit comport-il une forte quantité de données?
OUI
NON
Doit-on emmagasiner une grande quantité de résultats logiques?
OUI
NON
Utiliser "une logique cablée"

Figure 11

CHAPITRE **4** *Les langages d'ordinateurs Les appareils de décodage des langages en « binaire pur »*

Répétons ici, une fois de plus, que l'ordinateur ne connaît que le langage ***binaire pur*** encore appelé : ***binaire naturel*** qui se traduit par une succession de 0 et de 1 aussi bien pour recevoir des demandes que pour effectuer des opérations arithmétiques ou des fonctions logiques.

Si l'on imposait au programmeur d'élaborer en conséquence ses instructions à donner à l'ordinateur directement en binaire naturel, reconnaissons qu'après une succession de quelques milliers de 0 et de 1 alignés à la suite des uns et des autres, il aurait grande chance de se tromper et de trouver ce travail plutôt lassant.

Aussi a-t-on imaginé des ***formes de langages*** de traduction plus colorées mises à la portée du programmeur.

Ces langages sont introduits dans une machine, qui sera, elle-même chargée de les traduire en « binaire naturel » afin d'être digéré par l'ordinateur.

Le programme élaboré par le programmeur s'appelle : le ***programme source.***

Le programme source décodé en binaire naturel par la machine et introduit dans l'ordinateur va constituer : le ***programme objet.***

Le tableau suivant nous montre quelques-uns des langages de programmation dont nous donnerons pour chacun les définitions et les différences essentielles.

Dans ce tableau en (1) les programmes-source relativement courts sont ***codés*** soit en ***octal,*** en ***hexadécimal*** ou en ***B.C.D.*** (nombres décimaux codés en binaire). Nous expliquerons leur fonctionnement dans le chapitre suivant réservé au : ***système de calcul binaire.***

Tableau des principaux langages de programmation

<table>
<tr><th rowspan="2">Programme</th><th colspan="2" rowspan="2">Codage ou langage</th><th colspan="2">Appareil traducteur en langage machine</th></tr>
<tr><th>Interne à l'ordinateur</th><th>Externe à l'ordinateur</th></tr>
<tr><td rowspan="3">①
Court</td><td colspan="2">Octal</td><td colspan="2" rowspan="3"></td></tr>
<tr><td colspan="2">Hexadécimal</td></tr>
<tr><td colspan="2">B.C.D.</td></tr>
<tr><td rowspan="6">②
Long</td><td colspan="2">Mnémonique ou symbolique</td><td>Assembleur résident</td><td>Cross-Assembleur</td></tr>
<tr><td rowspan="4">Langage évolué</td><td>Algol</td><td colspan="2" rowspan="4">Compilateur</td></tr>
<tr><td>Fortran</td></tr>
<tr><td>Cobol</td></tr>
<tr><td>PL1-APL</td></tr>
<tr><td>Langage Interprété</td><td>Basic</td><td colspan="2">Interprète</td></tr>
</table>

En ② les programmes-source relativement longs sont traités, pour ne pas être fastidieux, soit en ***langage Mnémonique,*** encore appelé ***symbolique*** ou soit en ***langage évolué*** ou enfin en ***langage interprète.***

La machine chargée de définir le programme objet, à partir du programme source est appelée, lorsqu'il s'agit d'un langage mnémonique un ***assembleur*** et le programme spécial chargé de la traduction : le ***programme assembleur.*** La traduction constitue l'***assemblage.***

Lorsque la traduction en « binaire naturel » est exécutée directement par l'ordinateur auquel est destiné le programme, l'assembleur est souvent appelé ***assembleur résident.***

Lorsque la traduction en « binaire naturel » (code machine) est exécutée par un autre ordinateur plus puissant que celui auquel le programme est destiné, il est dénommé : ***cross assembleur.***

L'ordinateur ainsi utilisé, doit, dans ce cas, respecter les règles d'assemblage du destinataire.

A noter que les ***pseudo-instructions*** ou ***directives*** sont des ordres non directement exploités par l'ordinateur, mais ils peuvent cependant déclencher une séquence spéciale de l'assembleur, destinée à des opérations d'organisation dans l'ordinateur comme : réserver des zones en mémoire, définir des adresses, etc.

Enfin, il existe toute une série de variantes d'assembleurs, tel que :

Le ***macro assembleur*** dans lequel chaque instruction du langage source, définit une opération plus complète décomposée en une véritable suite de séquences d'instructions.

L'***assembleur translatable*** (relocatable assembleur) qui traite des adresses relatives et à l'opposé :

L'***assembleur absolu*** chargé de calculer les adresses absolues ou directes des informations dans les mémoires.

Lorsqu'il s'agit d'un ***langage évolué,*** la machine chargée de traduire le programme source en programme objet est appelée un ***compilateur.*** La traduction constitue : La ***compilation.***

Nous avons, dans notre tableau, cité quelques langages évolués à titre d'exemple. Voyons leur utilisation spécifique.

Pour les langages à vocation universelle : LE PL1 et l'APL.

Pour les langages à vocation scientifique : l'Algol et le Fortran.

Pour les langages orientés vers la résolution des problèmes de gestion : le Cobol.

Le langage mnémonique, comme les codages « octal », hexadécimal, BCD sont des ***langages élémentaires*** par opposition au langage évolué.

Ainsi, pour effectuer une simple opération, il faudra en langage mnémonique, ou ***assemblé*** toute une série de séquences d'instructions nécessitant déjà une certaine connaissance intime du fonctionnement de l'ordinateur, alors qu'en langage évolué, il suffira d'écrire sa demande, simplement d'une façon conforme au langage particulier utilisé.

Un micro-ordinateur comportant ***un microprocesseur*** ne peut à présent qu'employer un langage mnémonique.

Un exemple en est donné dans le chapitre 3 où ***FETCH*** et ***LOAD*** sont des exemples d'instructions symboliques.

Le langage d'assemblage employé demande un certain temps de programmation plus long que celui du langage évolué.

Par contre, il peut être élaboré d'une manière plus fine et nécessitera moins de place en mémoire.

Faut-il donc économiser le temps de programmation à l'aide d'un langage évolué ?

Ou bien encore économiser de la mémoire avec un langage d'assemblage ?

Aux vues de ces 2 affirmations, la réponse semble être que les micro-ordinateurs doivent être d'un prix de revient intéressant chaque fois qu'ils s'appliquent à des emplois spécialisés ne nécessitant qu'un seul ou encore quelques programmes simples et peu nombreux.

Le ***langage interprété (Basic)*** traduit dans ***l'interprète*** les instructions du programme source en programme objet une à une et procède pas à pas à leur exécution immédiate.

Le temps d'exécution du programme est donc obligatoirement plus long que celui d'un langage « assemblé » ou « compilé ».

Par contre, ce procédé convient parfaitement à **l'enseignement de la programmation,** puisque l'on peut, si on le désire, obtenir les résultats, après chaque étape d'instruction exécutée.

CHAPITRE 5 *Le calcul binaire Les codages : octal, hexadécimal et B.C.D.*

Afin d'effectuer les opérations arithmétiques, il est indispensable au premier chef de définir la façon d'écrire les nombres quelle que soit la base du système de numération adoptée.

Cette manière d'écrire les nombres se résume dans une ***formule*** que nous posons après l'avoir démontrée en prenant comme exemple un nombre du système à base 10 ***(système décimal).***

Soit le nombre de 3 chiffres : 825

Il peut s'écrire : $(8 \times 10^2) + (2 \times 10^1) + (5 \times 10^0)$

C'est-à-dire : 800 + 20 + 5

800 - ordre des centaines et 8 chiffre de 3e rang.

20 - ordre des dizaines et 2 chiffre de 2e rang.

5 - ordre des unités et 5 chiffre de 1er rang.

La base étant 10, la formule générale décomposée est :

$N = a.10^2 + b.10^1 + c.10^0$

En remplaçant maintenant :

Les coefficients a (des centaines), b (des dizaines) et c (des unités) par un coefficient général α doté d'un indice (n) représentant le rang le plus élevé du nombre.

Et la base 10, la lettre B. On parvient à la formule générale :

$$N = \alpha_n B^{n-1} + \alpha_{n-1} B^{n-2} \dots + \alpha_1 B^0$$

Cette formule peut alors s'appliquer à tous les nombres (N) quelle que soit la base (B) du système de numération choisi.

Dans la pratique courante de calcul, on a adopté le ***système décimal*** dans la plupart des pays du monde.

En informatique, pour des raisons de commodité de translation de calcul en signaux électriques travaillant par **tout ou rien** on a adopté le système binaire s'exprimant par les **bits** (0 et 1).

Comme l'on traite souvent, en micro-informatique notamment, de nombres à 4 éléments binaires, à 8 éléments binaires (8 bits = 1 octet) ou même à 16 éléments binaires on se sert des systèmes :

— BCD (Binary Coded Decimal)

— Octal (à base 8)

— Hexadécimal (à base 16)

Les 2 tableaux ci-après nous montrent les correspondances entre ces systèmes de numération utiles en informatique.

Au vu de ces 2 tableaux on peut observer :

— qu'il faut un groupe de 3 bits pour traduire en binaire tous les chiffres d'un nombre (0 à 7) lu dans le système octal ;

— qu'il faut un groupe de 4 bits pour écrire en binaire tous les chiffres

Tableau de correspondances entre le système décimal et les systèmes Octal, Hexadécimal et Binaire naturel

Valeur décimale	Codage binaire naturel	Codage Octal			Codage Hexadécimal	
		Valeur Octal	Codage en groupe binaire		Valeur Hexa.	Codage en groupe binaire
			gr. 2	**gr. 1**		
0	0	0	000	000	0	0000
1	1	1	000	001	1	0001
2	10	2	000	010	2	0010
3	11	3	000	011	3	0011
4	100	4	000	100	4	0100
5	101	5	000	101	5	0101
6	110	6	000	110	6	0110
7	111	7	000	111	7	0111
8	1000	10	001	000	8	1000
9	1001	11	001	001	9	1001
10	1010	12	001	010	A	1010
11	1011	13	001	011	B	1011
12	1100	14	001	100	C	1100
13	1101	15	001	101	D	1101
14	1110	16	001	110	E	1110
15	1111	17	001	111	F	1111
16	10000	20	010	000	10	10000

Tableau de correspondances entre le système décimal et les systèmes binaire naturel et B.C.D.

Valeur décimale	Codage binaire naturel	Système BCD Codage en groupes binaires	
		Dizaines	Unités
0	0	0 0 0 0	0 0 0 0
1	1	0 0 0 0	0 0 0 1
2	1 0	0 0 0 0	0 0 1 0
3	1 1	0 0 0 0	0 0 1 1
4	1 0 0	0 0 0 0	0 1 0 0
5	1 0 1	0 0 0 0	0 1 0 1
6	1 1 0	0 0 0 0	0 1 1 0
7	1 1 1	0 0 0 0	0 1 1 1
8	1 0 0 0	0 0 0 0	1 0 0 0
9	1 0 0 1	0 0 0 0	1 0 0 1
10	1 0 1 0	0 0 0 1	0 0 0 0
11	1 0 1 1	0 0 0 1	0 0 0 1

d'un nombre (0 à F) lu dans le système hexadécimal. Dans le système hexadécimal on représente les valeurs décimales de 10 à 15 par les premières lettres de l'alphabet : A, B, C, D, E, F ;

— Qu'il faut un groupe de 4 bits pour traduire en binaire chaque ordre (unités, dizaines, centaines, etc.) d'un nombre codé en B, C, D.

Nous donnerons d'ailleurs un exemple de codage binaire, octal et hexadécimal en fin de chapitre.

Voyons, comment, passer d'un système dans l'autre.

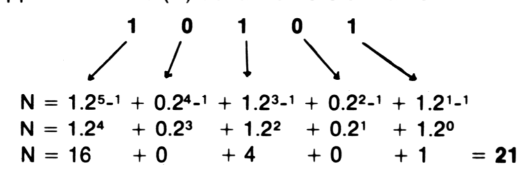
Soit, d'abord, le passage d'un nombre écrit dans le système binaire à un nombre correspondant écrit dans le système décimal.

Afin d'effectuer une telle opération, il suffit de se rappeler la formule précédente (N) que nous venons de développer.

On a par exemple le nombre binaire **10101**

qui s'écrit : **10101_2** (indice 2 du système binaire).

On développe la formule (N) de la manière suivante :



1 0 1 0 1

$$N = 1.2^{5-1} + 0.2^{4-1} + 1.2^{3-1} + 0.2^{2-1} + 1.2^{1-1}$$

$$N = 1.2^{4} + 0.2^{3} + 1.2^{2} + 0.2^{1} + 1.2^{0}$$

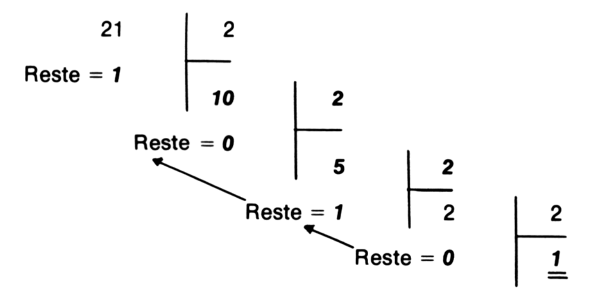
$$N = 16 + 0 + 4 + 0 + 1 = \mathbf{21}$$

21 est le nombre décimal correspondant au nombre binaire de 5 bits → 10101_2.

Soit, maintenant, le passage d'un nombre écrit dans le système décimal à un nombre correspondant écrit dans le système binaire.

La conversion peut s'effectuer de la façon suivante indiquée ci-dessous.

Conversion d'un nombre : Système décimal/système binaire



Pour écrire le nombre 21_{10} en système binaire, on remonte dans le sens de la flèche en prenant le dernier quotient (1) comme unité binaire la plus à gauche du nombre ainsi formé successivement avec les autres restes.

Soit $\mathbf{21_{10} = 10101_2}$

Jusqu'à présent nous n'avons discuté qu'à propos des ***nombres entiers.*** Voyons maintenant comment passer d'un système dans un autre avec des ***nombres fractionnaires.***

La formule permettant de définir la partie fractionnaire d'un nombre est :

$$N = \alpha_1 B^{-1} + \alpha_2 B^{-2} \ldots + \alpha_n B^{-n}$$

En se souvenant que B^{-1} est la représentation de $1/B_1$
B^{-2} est la représentation de : $1/B^2$
B^{-n} est la représentation de $1/B^n$

On va ainsi pouvoir passer d'un nombre écrit dans le système binaire au nombre correspondant dans le système décimal.

Soit à convertir $0{,}101_2$ en décimal

d'après la formule nous aurons :

$$0{,}101 = 1.2^{-1} + 0.2^{-2} + 1.2^{-3}$$
$$= 1/2^1 + 0 + 1/2^3$$
$$= 0{,}5 + 0 + 0{,}125 = 0{,}625_{10}$$

A noter que le nombre binaire $10101.101_2 = 21{,}625_{10}$

Afin d'opérer la conversion inverse de décimal fractionnaire à binaire, on effectue des multiplications successives :

— Soit à convertir $0{,}625_{10}$ en binaire naturel
0,625 x 2 = 1,250 → soit 1 de retenue → reste 0,250
0,250 x 2 = 0,500 → soit 0 de retenue → reste 0,500
0,500 x 2 = 1,000 → soit 1 de retenue → reste 0

Les retenues successives, en prenant comme unité la plus à gauche du nombre, la retenue de la 1re multiplication effectuée, forment le nombre binaire fractionnaire recherché.

Soit ici $0{,}625_{10} = 0.101_2$

Un nombre algébrique peut être négatif ou positif.

Sa valeur absolue, en nombre binaire doit donc être précédée d'un ***BIT de signe.*** Il existe plusieurs procédés de codage de ce signe.

Le plus souvent on emploie un 0 pour indiquer le signe (+) et un 1 pour indiquer le signe (—).

Ainsi le nombre 14_{10} qui s'écrit 1110_2 avec 4 bits donne :

$+14_{10} = 01110_2$ et $-14_{10} = 11110_2$

Nous allons maintenant passer en revue la façon dont on doit procéder pour effectuer dans le système binaire les 4 opérations principales : l'addition, la soustraction, la multiplication et la division.

1. L'addition

Ce procédé d'addition binaire est basé sur les règles suivantes :

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 + 1 de retenue

Exemple : 1101_2
Exemple : + 0101_2
= 10010_2

Dans notre exemple nous avons additionné deux nombres de 4 bits. Le résultat apparaît sous forme d'un nombre de 5 bits.

Si le centre de traitement n'est donné qu'en ***MOTS de 4 bits*** il y a ici un 5e ***bit de retenue*** que l'on peut généralement enregistrer dans l'ordinateur, dans un ***flip-flop*** spécial, prévu à cet effet, appelé par les Américains un ***carry*** ou bien encore un ***lien*** (Link).

2. La soustraction

Il y a plusieurs manières de l'effectuer. Nous étudierons celle qui nous semble en informatique la plus standard et qui consiste à opérer ***par complémentarité.***

On doit ajouter au 1er nombre binaire donné, le complément à 2, appelé ***complément vrai*** du deuxième nombre à soustraire.

Le ***complément à 2*** d'un nombre s'obtient en ajoutant (1_2) au ***complément à 1*** ou ***complément restreint*** de ce nombre.

Le ***complément à 1*** d'un nombre binaire s'obtient en changeant tous les 1 en 0 et tous les 0 en 1, de ce nombre.

Ainsi, soit à soustraire $\mathbf{0101_2}$ de $\mathbf{1010_2}$

Le complément à 1 de 0101_2 est → 1010_2

Le complément à 2 de 1010_2 est → 1011_2

Le résultat de la soustraction est donc :

$$\begin{array}{r} 1010_2 \\ +\ \ 1011_2 \\ \hline =10101_2 \end{array}$$

On élimine alors la dernière retenue à gauche de l'opération qui est (1). Le résultat de la soustraction binaire est 0101_2, ce qui se vérifie avec par exemple les correspondances des nombres en décimal :

$$\left.\begin{array}{l} 1010_2 \rightarrow 10_{10} \\ 0101_2 \rightarrow \ \ 5_{10} \end{array}\right\} 10_{10} - 5_{10} = 5_{10}$$

Quand le chiffre de retenue dans le calcul en binaire est comme ici (1), le résultat de la soustraction est un nombre de ***signe positif.***

S'il n'y a pas de retenue à gauche du nombre, ce que l'on peut signifier par un O, le résultat est un nombre de ***signe négatif.***

3. La multiplication

Nous étudierons deux manières simples de procéder :

a) La multiplication par ***additions successives.***

Soit en base 10 → 12 x 3 = 36 → en binaire : 1100 x 0011 que l'on pose :

```
   1100
+  1100
+  1100
-------
 100100
```

On additionne 3 fois le multiplicande c'est-à-dire que l'on effectue une somme contenant autant de fois le multiplicande qu'il y a d'unités au multiplicateur.

b) La multiplication par ***décalage et addition.***

On exécute la multiplication binaire comme on le ferait pour une multiplication décimale.

Exemple :

```
    1100
x     11
--------
    1100
   1100 .
--------
= 100100₂
```

4. La division

La division binaire est effectuée comme la division décimale :

```
  100100 | 11
— 00↓    |------
  100    | 01100
—  11    |
  0011   |
—   11   |
    0000 |
```

Nous venons d'effectuer les 4 opérations arithmétiques essentielles en « système binaire naturel ». C'est le seul codage ou ***programme objet*** ou bien encore ***code machine*** que l'ordinateur ou le micro-ordinateur connaisse.

Le programmeur peut par contre écrire un ***programme source*** plus simple en ***code octal*** ou en ***code hexadécimal,*** nous en donnons maintenant un exemple :

L'instruction « Clear 64 » dans le mini-ordinateur « PDP 11 » met à zéro le contenu de la position d'adresse 64.

Cette instruction se trouve codée sur 2 mots de 16 bits de la façon suivante : (en binaire naturel)

CLEAR — 0 000 101 000 011 111
64 — 0 000 000 001 000 000

En codage octal, on traduira chaque groupe de 3 bits à partir de la droite, ce qui donnera :

CLEAR	0	000	101	000	011	111	
	0	0	5	0	3	7	= **005037_8**
64	0	000	000	001	000	000	
	0	0	0	1	0	0	= **000100_8**

En codage hexadécimal, on traduira chaque groupe de 4 bits à partir de la droite, ce qui donnera :

CLEAR	0000	1010	0001	1111	
	0	A	1	F	= **$0A1F_{16}$**
64	0000	0000	0100	0000	
	0	0	4	0	= **0040_{16}**

La longueur d'un mot que peut traiter un ordinateur, peut parfois être insuffisante pour la précision requise.

Lorsqu'il y a dépassement du nombre de bits du mot de ***simple précision,*** on peut avoir recours pour traiter l'information à deux mots. On travaille alors en ***double précision.*** Cela pourrait être le cas du codage « BCD » pour exprimer deux mots de 4 bits représentant un nombre comportant des unités et des dizaines dans le système décimal.

Simple précision se dit encore : ***simple longueur***
Double précision se dit encore : ***double longueur***

Rien n'empêche de travailler par exemple en ***triple longueur.***

Lorsque l'on dit d'un microprocesseur qu'il est de 8 bits, cela signifie qu'il reçoit simultanément, autrement dit en parallèle, sur ses 8 entrées, les 8 bits de chaque mot composant un ***programme objet.*** Dans ce cas chaque mot peut donc comporter **2^8 = 256 combinaisons possibles de 1 et de 0,** ce qui va de suite nous suggérer une idée sur la précision ou ***la richesse de langage*** de cet ordinateur.

Enfin pour terminer ce chapitre sur le calcul binaire, amusons-nous à titre d'exercice de futur programmeur, à considérer un nombre binaire quelconque afin de le traduire dans les différents systèmes que nous venons d'étudier.

Soit le nombre binaire de 8 bits : $\mathbf{10010111_2}$.

S'il s'agit d'un **nombre binaire naturel** → il est égal à : $\mathbf{151_{10}}$

S'il s'agit d'un **nombre signé (Complt à 1)** → il est égal à : $\mathbf{-104_{10}}$

S'il s'agit d'un **nombre signé (Complt à 2)** → il est égal à : $\mathbf{-105_{10}}$

S'il s'agit d'un **nombre BCD** (2 groupes de 4 bits → il est égal à : $\mathbf{97_{10}}$

S'il s'agit d'un **nombre octal** (par groupe de 3 bits) → il est égal à : $\mathbf{227_8}$

S'il s'agit d'un **nombre hexadécimal** (par groupe de 4 bits) → il est égal à : $\mathbf{97_{16}}$

Enfin rien n'empêche que ce nombre binaire de 8 bits ne soit la traduction d'une instruction d'un programme source codé en langage mnémonique ou symbolique.

Exemple : Ce nombre binaire correspond dans le microprocesseur MC6800 de Motorola à l'instruction en langage mnémonique :
STAA qui signifie : emmagasiner dans l'accumulateur A. Le terme accumulateur fera l'objet d'une explication lors du chapitre consacré au microprocesseur.

CHAPITRE **6**

Les fonctions logiques

Le jeu d'instructions d'un microprocesseur comporte généralement, afin de résoudre les problèmes qui lui sont posés, des instructions de ***fonctions logiques.***

Ce sont bien souvent de simples ***opérateurs*** tels que les ***portes*** : ET, OU inclusif, ou exclusif, NAND, NOR... etc.

De plus, il est intéressant de se faire une idée sur la conception elle-même des circuits de ***mémorisation*** tels que : ***bascules bistables*** et ***registres*** participant à l'organisation d'un ordinateur ou d'un micro-ordinateur et structurant en partie son ***processeur*** ou son ***microprocesseur.***
C est la raison pour laquelle nous allons consacrer ce chapitre aux ***circuits logiques*** et à ***leurs fonctions.***

Les circuits logiques nécessaires aux fonctions correspondantes comportent des ***bornes d'entrées*** et des ***bornes de sorties.***

Le logicien applique des tensions aux bornes d'entrées et en recueille d'autres aux bornes de sorties qui sont fonction des combinaisons des tensions appliquées et des circuits employés.

Toutes ces tensions ne peuvent prendre que deux valeurs quantifiées.

1. Le ***niveau haut*** - symbole $\mathscr{H}$ correspondant au plus fort voltage.

2. Le ***niveau bas*** - Symbole $\mathscr{B}$ correspondant au plus faible voltage.

Entre ces 2 niveaux repérés, le logicien ne tient pas compte des dispersions physiques des tensions.

Dans son raisonnement, le logicien s'appuie sur 2 conventions symboliques opposées :

1re Convention logique

Il s'agit de la ***logique positive.***
Une ***affirmation*** se traduit par le « 1 » du système binaire et définit la présence du niveau « $\mathscr{H}$ ».

Une négation ou ***infirmation*** se traduit par le « 0 » du système binaire et définit la présence du niveau « $\mathscr{B}$ ».

2e convention logique

Il s'agit de la ***logique négative.***

Une affirmation se traduit par le « 0 » du système binaire et définit la présence du niveau « $\mathscr{H}$ ».

Une négation ou « infirmation » se traduit par le « 1 » du système binaire et définit la présence du niveau « $\mathscr{B}$ ».

Pour des raisons mnémotechniques bien évidentes, on a coutume aujourd'hui de raisonner de préférence en « logique positive ».

Afin d'illustrer nos dires, prenons, à titre d'exemple, la fonction logique : ***NON.***

Elle est définie, lorsque en appliquant à l'entrée, un signal « $\mathscr{H}$ », on recueille à la sortie un signal « $\mathscr{B}$ » ou bien encore en appliquant à l'entrée un signal « $\mathscr{B}$ » on recueille à la sortie un signal « $\mathscr{H}$ ».

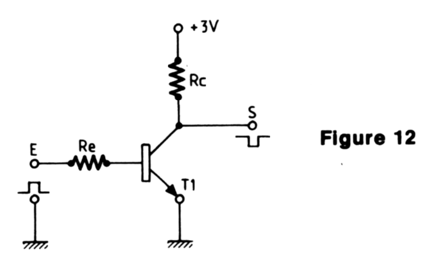


Figure 12

Afin de parvenir à ce résultat, imaginons (Fig. 12) un transistor que nous ferons alternativement travailler en ***régime saturé*** et en ***régime bloqué.*** Il est monté suivant la figure ci-après comportant en série dans son alimentation collecteur une résistance de charge « Rc ». D'autre part « Re » empêche de mettre la jonction émetteur-base en court-circuit.

1. On applique une impulsion positive de + 3 V en (E). Cette impulsion en (E) correspond au niveau ($\mathscr{H}$) égal à « 1 » en logique positive.

Le transistor (T_1) est saturé. Sur la borne « S » du circuit on recueille une tension de + 0,3 Volt égale à la tension de saturation de « T_1 ». Cette tension de + 0,3 Volt par rapport à la masse détermine le niveau « $\mathscr{B}$ » égal au « 0 » en logique positive.

2. On applique maintenant une tension nulle en « E », soit la tension masse de référence. Celle-ci représente le niveau « $\mathscr{B}$ » correspondant au niveau logique « 0 » dans le cas de la logique positive.

Le transistor (T_1) est bloqué. Sur la borne « S » on recueille une tension de + 3 Volts égale à la tension d'alimentation du transistor. Cette tension

représente le niveau « $\mathcal{H}$ », correspondant au niveau logique « 1 », dans le cas de la logique positive.

Nous pourrions raisonner, de la même manière dans le cas de la logique négative, mais les « 1 » et les « 0 » logique seraient inversés c'est-à-dire qu'à chaque niveau « $\mathcal{H}$ » correspondrait le « 0 » logique et à chaque niveau « $\mathcal{B}$ » correspondrait le « 1 » logique.

Le fabricant a l'habitude de résumer les caractéristiques de niveaux logiques de ses circuits dans ce que l'on appelle des ***tables de vérités.*** A titre d'exemple, nous présentons en figure 13, la table de vérités de la fonction « non ».

Fonction NON

Niveaux électriques		Table de vérités			
		Logique Positive		Logique Négative	
E	S	E	S	E	S
$\mathcal{H}$	$\mathcal{B}$	1	0	0	1
$\mathcal{B}$	$\mathcal{H}$	0	1	1	0

Figure 13

Il existe deux grandes classes de fonctions logiques :

1. La classe des ***fonctions logiques combinatoires.*** Dans cette classe, le signal recueilli à la borne « S » d'un circuit ne dépend exclusivement que de la combinaison des signaux appliqués au même instant aux bornes d'entrées (E) de ce circuit.

Nous verrons que l'***algèbre de Boole*** est d'un emploi particulièrement commode pour tous les circuits de logique combinatoire.

2. La classe des ***fonctions logiques séquentielles.***

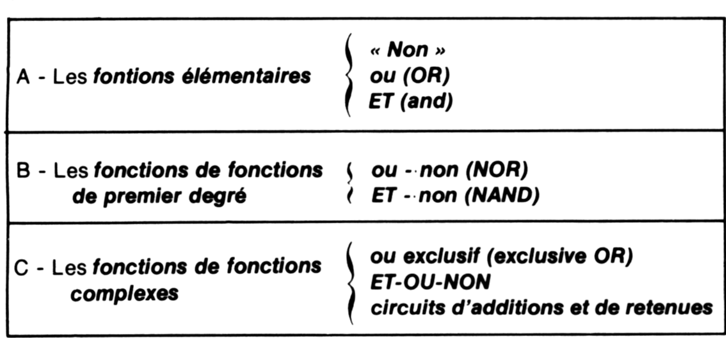
Le signal recueilli à la borne « S » d'un circuit ne dépend plus exclusivement de la combinaison des signaux appliqués à l'instant repère aux bornes d'entrée « E » mais également de l'état antérieur existant alors à l'application de ces dits signaux.

Il s'agit donc de fonctions qui possèdent une qualité de ***mémorisation*** des états antérieurs à l'instant considéré.

Ils tiennent compte de la ***séquence*** complète des données antérieures.

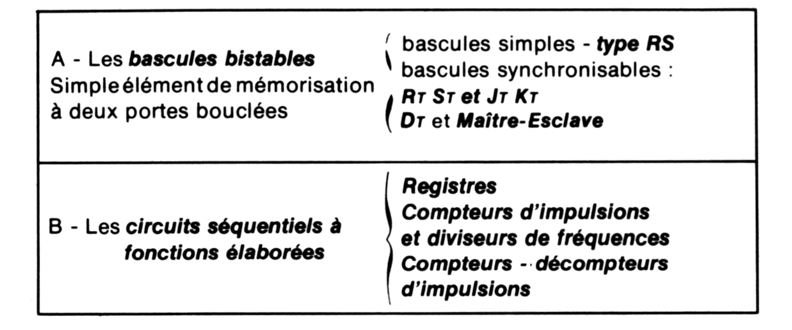
L'algèbre de Boole n'est alors guère utilisable et l'on raisonne à partir de : ***diagrammes de temps,*** de ***tables de vérités*** ou bien encore de : ***tables de phases*** où des colonnes permettent de distinguer les niveaux sur les bornes de sortie avant et après l'instant repère.

Les fonctions logiques combinatoires. peuvent se diviser en 3 grandes catégories hiérarchisées suivant leur degré de complexité :



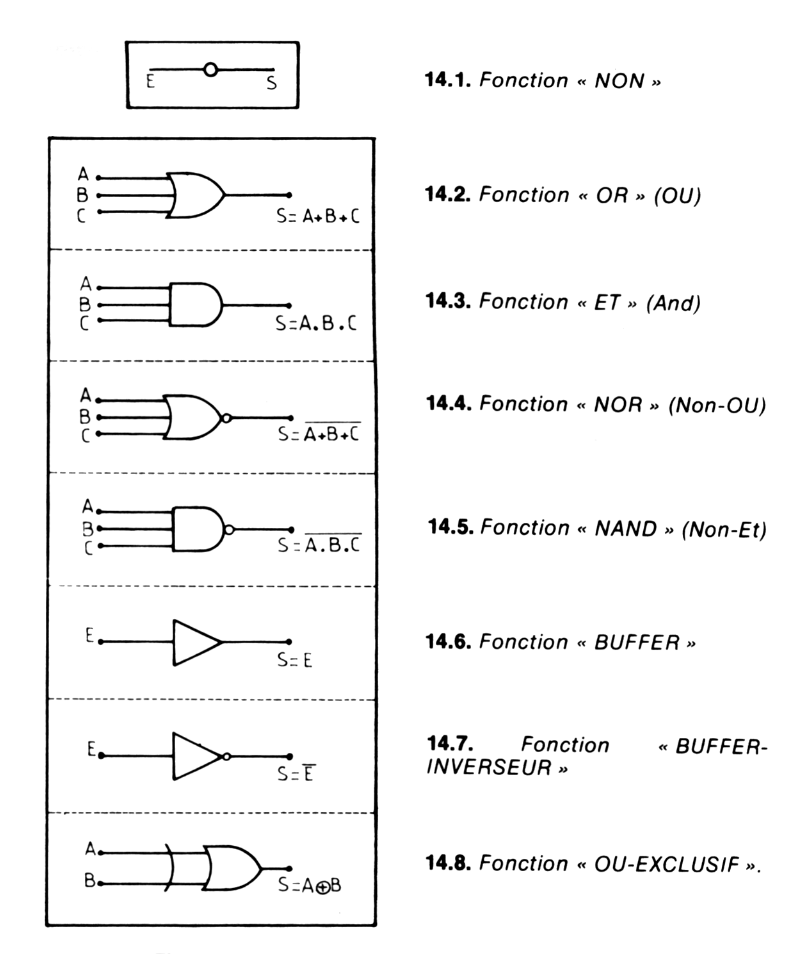
A - Les ***fontions élémentaires***	***« Non »*** ***ou (OR)*** ***ET (and)***
B - Les ***fonctions de fonctions de premier degré***	***ou - non (NOR)*** ***ET - non (NAND)***
C - Les ***fonctions de fonctions complexes***	***ou exclusif (exclusive OR)*** ***ET-OU-NON*** ***circuits d'additions et de retenues***

Les fonctions logiques séquentielles peuvent se diviser en 2 grandes catégories hiérarchisées suivant leur degré de complexité



A - Les ***bascules bistables*** Simple élément de mémorisation à deux portes bouclées	bascules simples - ***type RS*** bascules synchronisables : R_T S_T ***et*** J_T K_T D_T et ***Maître-Esclave***
B - Les ***circuits séquentiels à fonctions élaborées***	***Registres*** ***Compteurs d'impulsions*** ***et diviseurs de fréquences*** ***Compteurs - décompteurs d'impulsions***

Nous allons d'abord examiner *« les fonctions élémentaires de la logique combinatoire »*.



14.1. *Fonction « NON »*

14.2. *Fonction « OR » (OU)*

14.3. *Fonction « ET » (And)*

14.4. *Fonction « NOR » (Non-OU)*

14.5. *Fonction « NAND » (Non-Et)*

14.6. *Fonction « BUFFER »*

14.7. *Fonction « BUFFER-INVERSEUR »*

14.8. *Fonction « OU-EXCLUSIF ».*

Figure 14

1. Fonction « NON »

Nous en avons déjà donné la définition au cours de notre exposé. Cette fonction s'appelle encore une fonction dite ***fonction complémentée*** ou ***fonction inverse.***

En algèbre de Boole la fonction s'écrira : $\boxed{S = \bar{E}}$
et l'on prononcera : S = E ***barrée*** ou bien encore :
S = E ***complémentée.***

La figure 14-1 représente le symbolisme schématique utilisé ***(Normes MIL)*** pour représenter la fonction « Non ».

2. Fonction « OU » (OR)

Le circuit utilisé comporte plusieurs bornes d'entrées A, B, C.. etc et une seule borne de sortie « S ».

En logique positive, une affirmation en S (1 logique) sera recueillie chaque fois qu'au moins un signal de logique « 1 » sera appliqué sur l'une des bornes d'entrée A, B ou C du circuit.

En algèbre de Boole, l'équation de la fonction « OU »

s'écrit : $S = A + B + C$

et se lit : S = A *ou* B *ou* C

Le signe (+) indique le terme (OU)

La figure 14-2 représente le symbolisme schématique utilisé (normes MIL) pour représenter la fonction « OU ».

Dans le cas présent nous avons représenté un « opérateur » ou « Porte » à 3 entrées dont nous donnons en figure 15 la ***table de vérités*** de la fonction.

Fonction "OU"			
A	B	C	S
1	1	1	1
1	1	0	1
1	0	1	1
0	1	1	1
1	0	0	1
0	1	0	1
0	0	1	1
0	0	0	0

Figure 15

3. Fonction « ET » (And)

Le circuit utilisé comporte plusieurs bornes d'entrées A, B, C... etc. et une seule borne de sortie « S ».

En logique positive, une affirmation en S (1 logique) ne sera recueillie que lorsque sera appliquée sur toutes les bornes d'entrées A, B, C... etc. du circuit un signal de logique « 1 ».

En algèbre de Boole, l'équation de la fonction « ET » s'écrit :

$$S = A \bullet B \bullet C$$

et se lit : S = A et B et C

Le point (•) indique le terme « ET ».

La figure 14-3 représente le symbolisme schématique utilisé (Normes MIL) pour définir les fonctions « ET ».

Dans le cas présent nous avons représenté une porte « ET » à 3 entrées et nous donnons en figure 16 le tableau de la « table de vérités » de la fonction.

Fontion "ET"			
A	B	C	S
1	1	1	1
1	1	0	0
1	0	1	0
0	1	1	0
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	0

Figure 16

Examinons en second lieu, *les fonctions de fonctions de premier degré* de la logique combinatoire.

1. Fonction « OU-NON » (NOR)
2. Fonction « ET-NON » (NAND)

Les circuits utilisés ont la même implantation quant aux bornes d'entrées et borne de sortie que les circuits utilisés pour les fonctions « OU » et « ET ».

Leurs « tables de vérités » sont obtenues à partir de celles des fonctions « OU » et « ET » d'une part et de celles des fonctions « NON » d'autre part.

Autrement dit, il suffit dans les tables de vérités des fonctions « OU » et « ET » de remplacer d'une façon *systématique les valeurs de* la colonne S par leurs valeurs ***complément*** *c'est*-à-dire de remplacer tous les 1 logique *par des 0 et les 0 logique* par des 1 logiques.

En algèbre de Boole l'équation de la fonction « OU-NON » (NOR).

s'écrit : $S = \overline{A + B + C}$

et se lit : S = A ou B ou C complémenté

De même l'équation de la fonction « ET-NON » (Nand)

s'écrit : $S = \overline{A \bullet B \bullet C}$

et se lit : S = A et B et C complémenté

La figure 14-4 et la figure 14-5 représentent le symbolisme schématique utilisé (NORMES MIL) pour définir :

1. La fonction « OU-NON » (NOR)
2. La fonction « ET-NON » (NAND)

Les symboles schématisés rappellent donc, comme on le voit, la suite des 2 opérations successives (« OU » suivi de « NON ») et (« ET » suivi de « NON »).

On pourrait encore citer en marge des fonctions élémentaires et des fonctions de fonctions élémentaires, des circuits chargés uniquement d'amplifier les signaux utilisés lors de communications sur des lignes comportant des pertes importantes. Ces circuits sont appelés des circuits ***BUFFER*** (pour les amplificateurs simples) ou encore des circuits ***BUFFER INVERTER*** (Buffer inverseur) pour les amplificateurs inverseurs de signaux.

La figure 14-6 et la figure 14-7 représentent le symbolisme schématique utilisé pour désigner :

1. La fonction simple « Buffer »
2. La fonction « buffer inverter »

Les identités booléennes *(Théorème de de Morgan).*

Résumons cet exposé concernant la logique combinatoire par le tableau suivant :

1	2	3	4	5	6	7	8	9	10
A	B	$\bar{A}$	$\bar{B}$	$\bar{A}.\bar{B}$	$\bar{A}+\bar{B}$	$A.B$	$\overline{A.B}$	$A+B$	$\overline{A+B}$
1	1	0	0	0	0	1	0	1	0
0	0	1	1	1	1	0	1	0	1
1	0	0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	1	1	0

Ce tableau s'applique à une porte « **ET** » ou à une porte « **OU** » à deux entrées « **A et B** ».

— En **1 et 2** on affiche les 4 combinaisons possibles en ***binaire naturel*** des entrées « **A et B** ».

— En **3 et 4** on affiche les compléments de « **A et de B** ».

— En **5** on effectue la fonction logique « **ET** » avec les compléments de « **A et de B** ».

— **En 6** on effectue la fonction logique « **OU** » avec les compléments de « **A ou de B** ».

— **En 7** on effectue la fonction logique « **ET** » avec les nombres binaires des entrées « **A et B** ».

— **En 8** on affiche le complément de 7 (fonction **NON - ET**)

— En **9** on effectue la fonction logique « **OU** » avec les nombres binaires des entrées « **A où B** ».

— En **10** on affiche le complément de **9** (fonction **NON - OU**)

L'examen séparé de toutes les colonnes de 1 à 10 permet d'en déduire les identités booléennes suivantes formulées par de Morgan :

$\overline{A+B} = \bar{A}\cdot\bar{B}$ → colonnes 10 et 5
et $\overline{A\cdot B} = \bar{A} + \bar{B}$ → colonnes 8 et 6

Enfin pour terminer ce cours sur les fonctions de logique combinatoire élémentaires et les fonctions de fonctions de premier degré nous donnons en figure 17 une réalisation pratique. Il s'agit d'un circuit intégré logique de la famille **TTL 74** (technologie bipolaire : toutes logiques à transistors) appelé : **7400** comprenant **une quadruple porte Nand à 2 entrées** classique montée dans un boîtier **DIL** (Dual in line) dont les connections sont au nombre de **14**, alignées sur 2 lignes parallèles (2 x 7) et dont le boîtier est une encapsulation plastique et porte la référence normalisée : **TO116.**

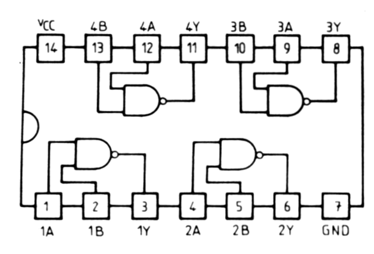


Figure 17 - Boîtier TO116

Examinons pour terminer cette étude sur les fonctions de logique combinatoire, *« les fonctions de fonctions complexes »*.

1. Fonction « OU exclusif » (OR exclusive)

La fonction « OU » que nous avons étudiée jusqu'à présent est encore appelée ***fonction ou inclusif*** c'est-à-dire que dans une porte à 2 entrées A et B → S = 1 si au moins une des entrées est portée au niveau logique « 1 » ou même encore si les 2 entrées sont portées au niveau logique « 1 ».

Dans la fonction ***ou exclusif*** → S = 1 si au moins 1 des entrées A ou B est portée au niveau logique « 1 » à l'exclusion de toutes les entrées portées au niveau logique « 1 » pour lesquelles la borne de sortie de la porte se trouve être alors au niveau logique « 0 ».

La figure 18 montre la table de vérités de la fonction logique « OU exclusif ».

Il apparaît en lisant cette table que lorsque

S = 1 → on a : A ≠ B
et S = 0 → on a : A = B

On imagine donc *une première application* de la fonction « OU exclusif ». Cette fonction peut servir de **comparateurs de signaux logiques.**

Fonction "ou exclusif"		
A	B	S
1	1	0
0	1	1
1	0	1
0	0	0

Figure 18

L'opérateur « OU exclusif » permet de vérifier les ***ETATS de coïncidence*** ou d'***"ANTI-coïncidence*** de signaux logiques.

Une deuxième application que nous décrivons en fin d'exposé est celle d'élément constitutif dans la conception d'un ***additionneur binaire.***

La figure 14-8 représente d'une manière symbolique le schéma suivant les normes MIL d'un circuit « OU exclusif ».

La fonction s'écrit : $S = A \oplus B$

et se lit : S = A ou B exclusif

Le signe (+) entouré d'un ◯ indique la fonction écrite « OU exclusif ». En algèbre de Boole il est démontré que :

$$S = A \oplus B = A \bullet \bar{B} + B \bullet \bar{A}$$

2 . Fonction « ET-OU-NON »

Cette fonction de fonctions combinatoire complexe est généralement obtenue à l'aide d'un circuit constitué par la combinaison d'opérateurs « And », « Or » et « Non ».

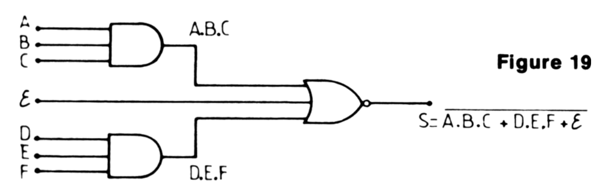


Figure 19

A titre d'exemple, nous figurons un schéma possible de réalisation de cette fonction (voir Fig. n° 19).

3. Fonction d'« addition et de retenue binaire »

Il s'agit ici d'une véritable « logique câblée » comprenant 2 circuits : ***demi-additionneur*** formés chacun d'un opérateur « OU exclusif » et d'une porte « And ». Nous ne donnons pas ici l'explication de fonctionnement de ce circuit qui dépasse le cadre de cet exposé. Disons seulement que nous avons un nombre binaire A_n à additionner à un autre nombre binaire B_n (voir Fig. n° 20)

L'addition avec les nombres précédents A_{n-1} et B_{n-1} a pu donner lieu à une retenue R_{n-1} à additionner également à $A_n + B_n$. Le résultat de l'addition est S_n et peut comporter une retenue R_n.

Nous allons aborder maintenant les ***fonctions logiques séquentielles.*** Nous étudierons seulement les « bascules » ou ***flip-flop*** ainsi que les

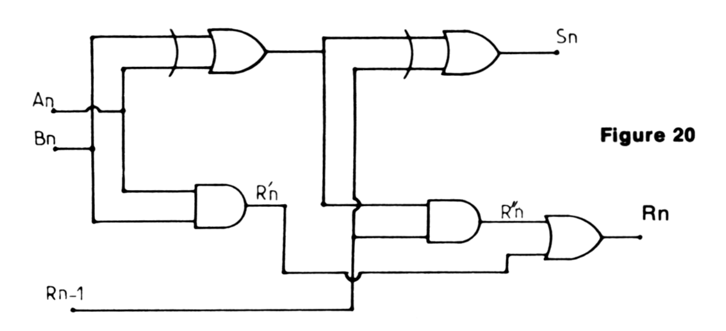


Figure 20

registres afin de se donner une idée élémentaire concernant la structure des ***mémoires.***

Nous commencerons cet exposé avec les ***bascules bistables.*** On examinera, successivement les bascules simples ***RS*** puis les bascules synchronisables : ***$R_T S_T$ - $J_T K_T$ - D_T*** et ***Maître-esclave.***

1. Bascules « RS »

Les plus simples sont constituées à l'aide de deux portes « NOR » ou « NAND » bouclées sur elles-mêmes (voir Fig. n° 21).

Ces bascules appelées encore « flip-flop » sont du type RS (***Reset*** et ***Set***). Leur représentation symbolique (Normes MIL) est celle d'un rectangle comme l'indique la figure 22 montrant également la table de vérités.

Une bascule est dans l'état « 1 »
ou l'**état actif** lorsque
$Q = 1$ (niveau $\mathcal{H}$ en logique positive)
$\overline{Q} = 0$
Elle est dans l'Etat « 0 »
ou l'**état inactif** lorsque
$Q = 0$ (niveau $\mathcal{B}$ en logique positive)
$\overline{Q} = 1$

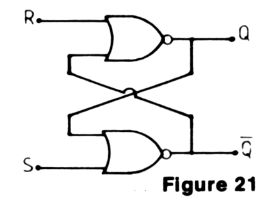


Figure 21

2. Bascules synchronisables

Tout d'abord, soit les bascules « $R_T S_T$ » et « $J_T K_T$ ».

La figure 23 donne leur représentation symbolique (Normes MIL).

J_T et K_T (ou R_T et S_T) sont les bornes d'entrées où l'on applique les états logiques devant définir l'état futur de la bascule.

« T ou C_p ». C'est la borne où l'on applique le signal de synchronisation qui donne **l'ordre ultime de basculement.** Ce signal est appelé : ***signal***

N° Etats	R	S	Q	$\bar{Q}$	Observations sur sorties	Schéma symbolique
1	$\mathscr{H}$	$\mathscr{H}$			1. Etat indéterminé si l'état précédent était 2	R, S → Q, $\bar{Q}$
2	$\mathscr{B}$	$\mathscr{B}$				
3	$\mathscr{B}$	$\mathscr{H}$	$\mathscr{B}$	$\mathscr{H}$	2. Etat déterminé mais sans changement si l'état précédent était 3 ou 4	
4	$\mathscr{H}$	$\mathscr{B}$	$\mathscr{H}$	$\mathscr{B}$		
3	$\mathscr{B}$	$\mathscr{H}$	$\mathscr{H}$	$\mathscr{B}$		R, S (entrées inversées) → Q, $\bar{Q}$
4	$\mathscr{H}$	$\mathscr{B}$	$\mathscr{B}$	$\mathscr{H}$		

Figure 22

d'horloge. « T » signifie : ***Trigger*** et C_p = ***Clock pulse « Q*** et $\bar{Q}$ **»** sont les bornes de sorties. Les niveaux obtenus de Q et de $\bar{Q}$ sont donc toujours complémentaires.

R_d et S_d sont dites : **Bornes de forçage** car elles imposent l'Etat de la bascule instantanément et indépendamment de « T ».

Rd
J_T
Q
T
K_T
$\bar{Q}$
Sd

Figure 23

Pour établir la **Table de phases** indiquant le fonctionnement d'une bascule synchronisable, soit R_TS_T ou J_TK_T, il faut considérer le temps t_n avant l'application de l'ordre ultime de basculement et le temps $t_n + 1$ après exécution de cet ordre.

En regardant les « tables de phases » (Fig. 24) nous observons que la différence entre les bascules R_TS_T et J_TK_T réside dans le fait que : **L'indétermination d'états en sortie est levée dans la bascule J_TK_T lorsque l'on applique, sur les 2 entrées, 2 niveaux « 1 » en logique positive.** De plus on constate dans ce cas un basculement des niveaux de sorties dans la bascule J_TK_T. Il n'y a pas de changement d'état pour des entrées auxquelles on applique un niveau logique « O » aussi bien dans les bascules R_TS_T que dans les bascules J_TK_T.

Examinons ensuite, les bascules du type : ***D_T***.

Il s'agit d'une bascule synchronisable simplifiée, puisqu'elle ne comporte qu'une seule entrée.

Tables de Phases des bascules R_TS_T et J_TK_T

Type de bascules	RT_{tn}	ST_{tn}	Q_{tn+1}	$\overline{Q}_{tn+1}$
Bascule R_TS_T	1	1	Etats indéterminés	
	1	0	1	0
	0	1	0	1
	0	0	Q_{tn}	$\overline{Q}_{tn}$
Bascule J_TK_T	1	1	$\overline{Q}_{tn}$	Q_{tn}
	1	0	1	0
	0	1	0	1
	0	0	Q_{tn}	$\overline{Q}_{tn}$

Figure 24

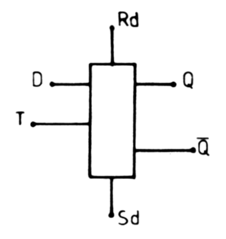


TABLE de PHASES – BASCULE Dt

Dtn	Qtn+1	$\overline{Q}$tn+1
0	0	1
1	1	0

Figure 25

Nous représentons en figure 25, son schéma symbolique et nous indiquons sa table de phases.

Voyons enfin la bascule du type : ***Maître-esclave (Master-slave).***

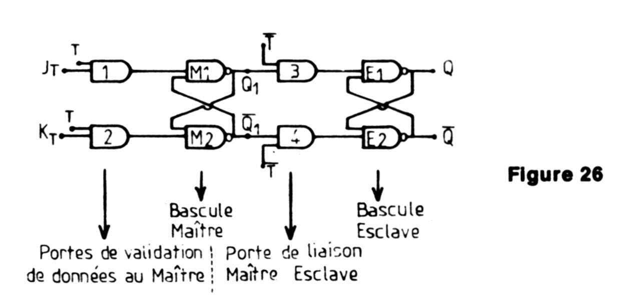


Figure 26

Le schéma de la figure 26 nous montre la configuration sophistiquée de ce genre de bascule.

— Les circuits M_1 et M_2 définissent : la ***Bascule maître***

— Les circuits E_1 et E_2 définissent : la ***bascule esclave***

— Les portes 1, 2, 3, 4 sont dites ***ouvertes*** lorsqu'elles transmettent à leur sortie l'information présente à leur entrée. Dans le cas contraire, elles sont dites ***fermées.***

Si $T = \mathscr{B}$ et $\overline{T} = \mathscr{H} \rightarrow$ Les portes 3 et 4 sont ouvertes. Le maître impose les niveaux logiques $\overline{Q}_1$ et Q_1 à l'esclave.

Les portes 1 et 2, sont fermées. La bascule maître est ***verrouillée*** et ne perçoit pas les instructions données en J_T et K_T.

Si $T = \mathscr{H}$ et $\overline{T} = \mathscr{B} \rightarrow$ Les portes 3 et 4 sont fermées. La bascule esclave se trouve isolée du maître, elle demeure dans l'état imposé précédemment par le maître.

Les portes 1 et 2 sont ouvertes. Les informations sur J_T et K_T imposent au maître son nouvel état.

Modes de synchronisation des bascules (voir figure 26 bis)

Il existe deux possibilités de synchronisation des bascules :

1. *Les bascules à fronts.* Elles sont synchronisées soit sur le front ascendant, ou soit sur le front descendant du signal d'horloge.

Ce mode de fonctionnement est employé dans les bascules rapides et de ce fait, il est nécessaire que le front de synchronisation du signal d'horloge ait une raideur $\frac{\Delta v}{\Delta t}$ suffisante.

BASCULES A FRONTS
SYNCHRO. SUR FRONTS ASCENDANTS
SYNCHRO. SUR FRONTS DESCENDANTS
BASCULES A NIVEAUX
1
Horloge
0
Pente active $\frac{\Delta V}{\Delta t}$ du front ascendant
Pente active $\frac{\Delta V}{\Delta t}$ du front descendant
50% t_f
Seuil de basculement
1
Signal de sortie
0
50% t_r
T_{PLH} (retard au basculement)
T_{PHL}

Figure 26 bis

La rapidité du front de déclenchement du signal d'horloge peut être obtenue à l'aide d'un circuit différentiateur.

2. *Les bascules à niveaux*. Elles sont synchronisées lorsque le niveau du signal d'horloge s'abaissant atteint un seuil déterminé.

Ce mode de fonctionnement est employé seulement dans les bascules n'exigeant pas une grande rapidité. C'est souvent le cas des bascules « Maître-Esclave ».

Après l'examen des bascules simples Rs et des bascules synchronisables nous allons aborder l'étude *des registres* faisant parti de la panoplie des ***circuits séquentiels à fonctions élaborées.***

Les registres sont des organes à mémoire qui peuvent être divisés en 3 parties distinctes :

a) **Les organes d'entrées** qui sont ***validées*** grâce à un ***ordre d'inscription*** (ou ***ordre d'écriture***). La validation (par exemple un 1 logique) autorise l'écriture d'une information dans la mémoire.

b) ***Les organes de mémorisation.*** Chaque bascule peut par exemple conserver « un bit ». (Elle se maintient dans un état donné à un moment donné).

c) **Les organes de sorties** qui sont ***validées*** grâce à un ***ordre de lecture.*** La validation (par exemple un 1 logique) autorise la lecture d'une information emmagasinée dans chaque bascule correspondante de la mémoire.

On peut distinguer suivant leur structure câblée 4 sortes de registres.

a) Les registres à écriture et à lecture des bits ***en parallèle***

b) Les registres à écriture et à lecture des bits ***en série***

c) Les registres à écriture des bits ***en série*** et lecture ***en parallèle.***

d) Les registres à écriture des bits ***en parallèle*** et lecture ***en série.***

A titre d'exemple et pour mieux comprendre ce que nous venons de dire sur les registres, nous donnons en figures 27 et 28 deux schémas de réalisations simples de registres de structure câblées différentes.

1. Schéma d'un registre à écriture et lecture des bits « en série » (Voir Fig. n° 27).

Dans ce registre les bascules B_1, B_2 et B_3 sont câblées en série. A tout moment on peut mettre les bascules à « 0 » avant de procéder à une

Registre à écriture et lecture des bits « en série »

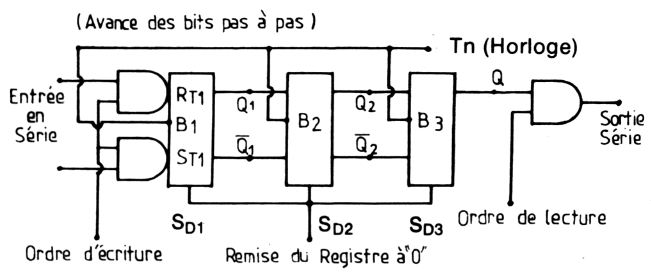


Figure 27

Registre à écriture et lecture des bits « en parallèle »

E1
E2
E3
E4
R1 B1 S1
R2 B2 S2
R3 B3 S3
R4 B4 S4
Q1
Q2
Q3
Q4
S1
S2
S3
S4
Ordre d'écriture
Ordre de lecture

Figure 28

nouvelle ***inscription*** à l'aide d'un ***ordre de remise du registre à zéro*** appliqué sur les bornes de forçage SD1, SD2, SD3 de chacune des bascules du registre.

Sur l'entrée série on envoie alors le 1er bit d'information ainsi que l'ordre d'écriture. Celui-ci est imposé dans la bascule B_1 au moment du ***temps de déclenchement*** du « signal d'horloge » T_n.

Au temps T_{n+1}, le 1er bit est « ***adressé*** » dans la bascule B_2 et si l'ordre d'écriture est donné, on peut également adresser dans B_1 un 2e bit d'information, au temps T_{n+2} on avance encore à nouveau d'un cran les bits d'informations.

— Le 1er bit se retrouve alors ***stocké*** en B_3 (MSB).

— Le 2e bit en B_2 et le 3e bit en B_1 (LSB).

Grâce à cette avance des bits ***pas-à-pas*** on a ainsi formé un mot de 3 bits.

Si l'on intime alors au registre « l'ordre de lecture », on pourra lire ainsi sur la sortie, successivement en série les 3 bits du mot aux temps : T_{n+3}, T_{n+4} et T_{n+5}, la lecture étant assurée par les déclenchements successifs de signaux d'horloge.

2. Schéma d'un registre à écriture et lecture des bits « en parallèle » (Voir Fig. n° 28).

Il s'agit ici d'un registre-mémoire pour mots de 4 bits à écriture et à lecture parallèles. L'ordre d'inscription et l'ordre de lecture s'effectuent donc simultanément sur les 4 bits qui composent chaque mot de cette mémoire.

Il est aisé de se rendre compte qu'un registre à mémoire est plus rapide à lire ou à inscrire quand les bits qui composent ses mots sont à accès parallèle, la lecture ou l'inscription s'effectuant pour tous les bits d'un mot simultanément c'est-à-dire au même temps t_n.

Nous avons ainsi examiné les principaux registres des circuits séquentiels à fonctions élaborées. Nous avons cité parmi ces fonctions élaborées d'autres circuits que les registres mais nous ne les étudierons pas ici car ils ne demeurent pas nécessaires à connaître pour notre cours d'initiation sur les micro-ordinateurs. De nombreux ouvrages ont d'ailleurs été édités à ce sujet et nous invitons nos lecteurs curieux à bien vouloir s'y reporter.

CHAPITRE **7**

La technologie des microprocesseurs

Le microprocesseur, comme nous l'avons déjà vu a pu prendre naissance grâce au développement des technologies de diffusion appliquées aux circuits intégrés ***LSI*** (large scale integration).

Les technologies les mieux adaptées à ces genres de circuits sont sans aucun doute les technologies ***MOS (N-MOS*** ou ***P-MOS)*** et ***C-MOS.***

On compte actuellement sur le marché plus de 50 microprocesseurs diffusés selon ces procédés de fabrication.

Ces technologies permettent à tout microprocesseur d'exécuter des additions dans un temps de l'ordre de 1 à 2 microsecondes.

La technologie N-MOS est un peu plus rapide que la P-MOS, les porteurs mobiles N étant moins lourds que les porteurs P.

Les consommations de puissance en cours de fonctionnement sont très faibles et la surface occupée par un transistor diffusé est infime, ce qui donne la possibilité d'intégrer, sans échauffement prohibitif toutes les fonctions d'un microprocesseur sur une seule pastille de silicium de quelques mm^2 de surface.

Quelques essais avaient auparavant été réalisés avec une technologie ***TTL/S*** (Logique tout à transistors et diodes Schottky) qui a l'avantage de la rapidité d'exécution des opérations, une addition pouvant s'effectuer en moins de 1/10e de microseconde.

Cependant cette technologie consomme davantage de puissance en cours de fonctionnement et l'on est généralement obligé de composer plusieurs circuits intégrés (au moins 2) si l'on désire définir toutes les fonctions essentielles d'un microprocesseur.

Pour cette raison la technologie TTL/S est aujourd'hui dans la fabrication des microprocesseurs pratiquement abandonnée.

Afin d'obtenir des vitesses d'exécution élevées, Motorola a pensé développer un microprocesseur en technologie ***ECL*** (Emitter Coupled Logic). Logique à « niveau H » non saturé.

Enfin quelques microprocesseurs vont apparaître sur le marché. Ils font appel à la technologie appelée ***I^2L*** (integrated injection Logic). La rapidité d'exécution est moyenne, par contre la puissance consommée est faible et le processus de fabrication est simple (peu de masques à utiliser en diffusion).

La plupart des microprocesseurs actuels sont des microprocesseurs 8 bits, ce qui signifie qu'ils acceptent des mots formés de 8 bits en parallèle ou *OCTETS*.

Il en existe en moins grand nombre de 4 bits, de 12 bits et de 16 bits. *Leur capacité d'adressage en mémoire* est actuellement pour la plupart d'entre eux de 64K octets.

Ils sont encapsulés dans un boîtier ***DIL*** (Dual In Line) comportant suivant les modèles de ***16 à 64 broches d'accès.***

La figure 29 nous montre la photographie de l'un d'eux.

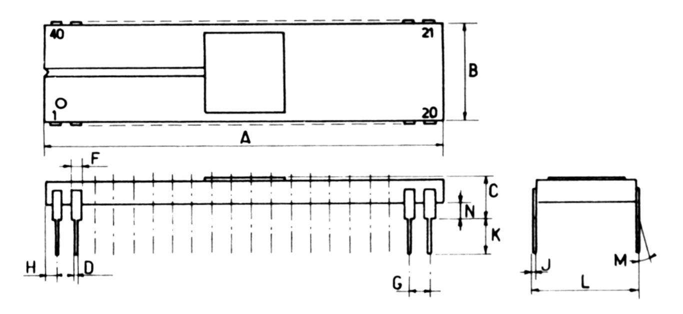
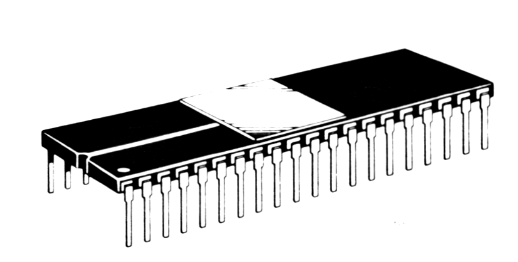


Figure 29

DIM	MILLIMETRES MIN	MILLIMETRES MAX
A	50.29	51.31
B	14.86	15.62
C	2.54	4.19
D	0.38	0.53
F	0.76	1.40
G	2.54 BSC	
H	0.76	1.78
J	0.20	0.33
K	2.54	4.19
L	14.60	15.37
M	–	10°
N	0.51	1.52



Nous nous proposons maintenant dans ce chapitre d'expliquer la physique des technologies MOS (N-MOS et P-MOS). CMOS ainsi que donner une idée concernant les technologies TTL/S, ECL et I²L. Nous dresserons enfin, à titre documentaire, suivant la technologie utilisée, le tableau provisoire des microprocesseurs actuellement commercialisés.

Procédé de fabrication d'un transistor MOS

On dispose d'un ***substrat de Si monocristallin dopé P.*** On fabrique alors un transistor ***MOS canal N.*** On place au-dessus du substrat un ***diélectrique*** constitué par de l'oxyde de silicium (SiO^2).

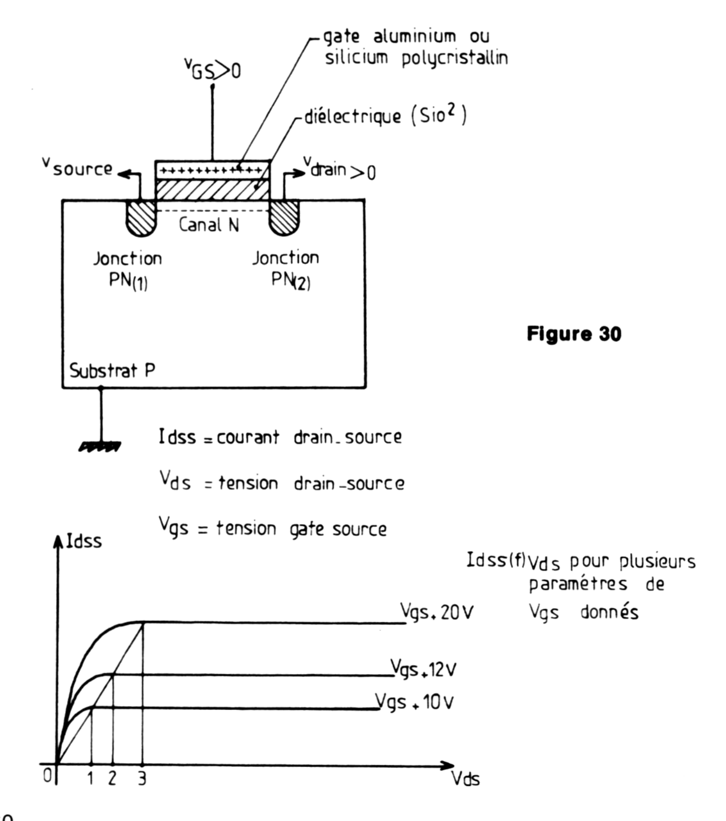


Figure 30

Au-dessus de l'oxyde, on dépose un contact qui formera la ***gate.*** Elle est constituée soit avec de l'aluminium, soit avec du silicium polycristallin.

L'ensemble : Substrat P, diélectrique et gate compose un ***condensateur.***

Si l'on applique sur l'armature « gate » une tension $V_{GS} > 0$ ***(tension gate-source)*** c'est-à-dire plus grande que la tension masse du substrat qui est dopé P, on doit observer :

— Sur le substrat P, face au diélectrique (SiO^2) ***des charges d'électrons N.***

— Sur la gate face au diélectrique ***des charges de trous positifs P.*** On a alors affaire à un condensateur (c) chargé.

Si de part et d'autre de ce condensateur, on diffuse ***2 jonctions identiques PN(1) et PN(2)*** et si l'on appelle *la jonction PN(1) la* ***Source*** et la jonction PN(2) le ***Drain,*** en polarisant positivement le drain par rapport au substrat et à la source, ou aura création d'un courant drain ***Idss*** qui dépendra avant tout de la tension ***VGS*** appliquée sur la gate du transistor MOS. C'est ce que traduit la courbe figure 30 du courant drain Idss en fonction de ***Vds*** en prenant V_{GS} (tension de la gate) comme paramètre. En effet plus V_{GS} est positif, plus la charge du condensateur MOS est grande et plus la tension positive de drain peut capter d'électrons du Canal N.

On peut également partir d'un ***substrat N.*** La charge entre source et drain sera alors constituée de trous positifs. V_{GS} et V_{ds} par rapport au substrat et à la source seront alimentés par des tensions négatives constituant face au diélectrique SiO^2 sur la gate, une réserve importante d'électrons.

Les trous positifs circuleront, cette fois, dans le canal P entre la sourceet le drain.

Nous aurons constitué un transistor ***MOS canal P.***

Procédé de fabrication, de 2 transistors MOS complémentaires (C-MOS).

La figure nous montre qu'il s'agit de ***Doper*** dans un substrat P une zone A suffisante avec des impuretés N en ***densité de concentration*** supérieure à celle du dopage P originel pour former ensuite un transistor Canal N dans le substrat P d'origine et un transistor canal P dans le nouveau substrat N constitué dans la zone A (voir Fig. n° 31).

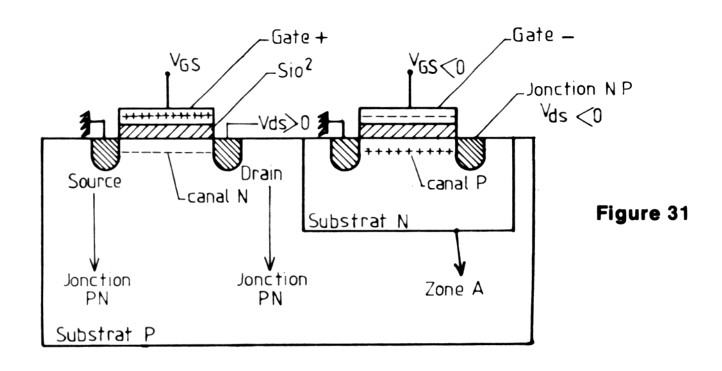


Figure 31

Applications pratiques de ces technologies MOS et CMOS aux systèmes microprocesseur.

1. Utilisation des transistors MOS dans les mémoires

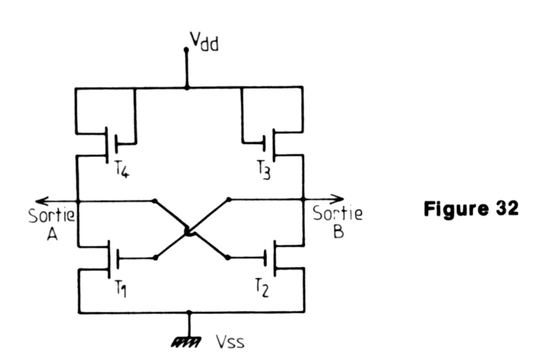


Figure 32

D'abord en mémoire ***RAM-statique.*** Nous représentons ici un ***Flip-Flop*** (MOS). T_1 et T_2 sont chargés par T_3 et T_4 dont gate et source sont court-circuitées et tiennent lieu de ***Résistance de charge.*** Une bascule ainsi montée est à ***2 états stables*** possibles (niveau haut ou bien niveau bas). Le flip-flop, tant qu'il est alimenté peut conserver une information (Bit 0 ou Bit 1) ***indéfiniment.*** C'est la raison pour laquelle, il définit un ***point mémoire*** d'une RAM appelée statique.

Ensuite on peut également utiliser la technologie MOS dans le cas des mémoires ***RAM Dynamique.*** Cette fois-ci on utilise l'effet de capacité

gate-substrat des transistors MOS, comme élément de mémorisation d'un état (bit 0 ou bit 1). Toutefois la charge du condensateur se dégradant toutes les 5 à 10 millisecondes, il faut ***rafraîchir*** la mémoire, c'est-à-dire lui adresser périodiquement, par exemple toutes les 2 millisecondes une charge pour compenser les méfaits du courant de fuite (pertes de la capacité gate-substrat).

On définit ainsi une mémoire RAM dynamique dont l'avantage est de n'utiliser que 3 transistors au lieu de 4 pour définir un ***point mémoire*** identique à celui d'un flip-flop (voir Fig. n° 33).

Dans les mémoires à forte capacité, par exemple les RAM dynamiques 16K bits ont fait encore mieux en n'utilisant qu'un seul transistor MOS et en renforçant la capacité du transistor MOS (T_1 de la figure n° 34) par des capacités physiques de mémoire supplémentaires. Ce procédé fait encore gagner de la place et permet ainsi d'inscrire un plus grand nombre de jonctions sur une même surface de pastille de silicium.

2. Utilisation des transistors C-MOS dans la fonction logique « Non »

La figure n° 35 nous montre le principe de câblage d'un ***inverseur CMOS*** effectuant ainsi la fonction logique « NON ». La tension d'alimentation V_{DD} à l'entrée V_I met en conduction le transistor à canal N et bloque le transistor à canal P, si bien que la tension de sortie est égale à V_{SS}. De la même manière, une tension d'entrée égale à V_{SS} rend conducteur le transistor à canal P et bloque le transistor à canal N, si bien que la tension de sortie est égale à V_{DD}. Le circuit se comporte donc bien comme celui d'un inverseur pour lequel l'excursion de tension en sortie est égale à celle de la tension d'entrée. Dans ce montage à ***alimentation de tension série,*** suivant le niveau appliqué à l'entrée V_I, un des 2 transistors est prêt à débiter tandis que l'autre, mis à part son courant de fuite, demeure bloqué.

Le montage ne débite en somme, vraiment que pendant la période de basculement des niveaux d'entrée, la consommation de puissance d'un tel circuit est donc extrêmement faible.

La caractéristique de transfert (Fig. n° 36) d'un inverseur C-MOS que nous représentons ici, indique la variation du niveau de sortie V_0 en fonction de la tension d'alimentation appliquée Vi.

V_{TR} est la ***tension de transition*** pour laquelle s'opère le transfert de niveau de tension dans le circuit.

Protection des circuits d'entrée (Fig. n° 37). Les circuits C-MOS ont ***une très grande impédance d'entrée (10^{15} Ω)*** et durant leur manipulation ou bien leur contrôle, une charge statique d'énergie faible peut suffire à créer à l'entrée une haute tension capable de détruire un oxyde de gate

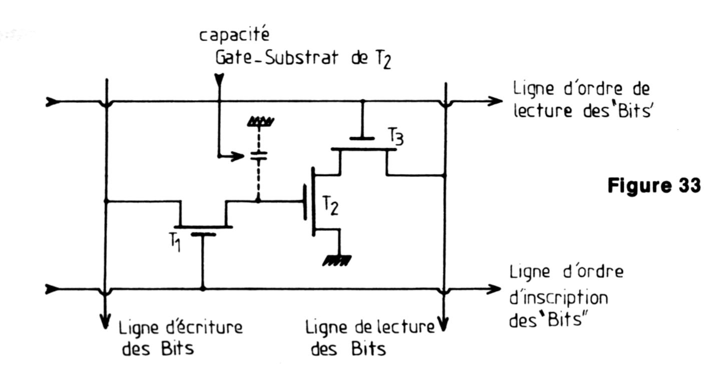


Figure 33

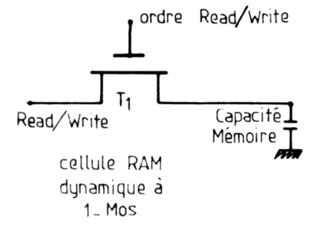


Figure 34

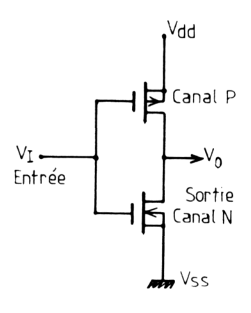


Figure 35

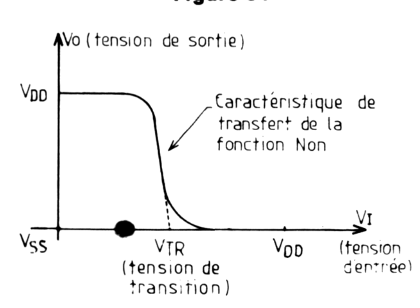


Figure 36

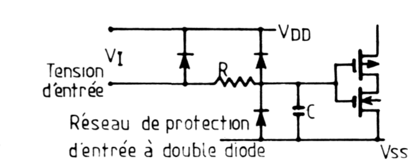


Figure 37

non protégé. Il est donc nécessaire de prévoir sur tous les circuits intégrés C-MOS ***un circuit de protection d'entrée*** tel que nous l'avons figuré. Une tension d'entrée parasite est atténuée par le filtre RC et les diodes de protection écrêtent les amplitudes.

Fonctionnement de la TTL/S (Fig. n° 38)

Nous avons vu que l'avantage essentiel de cette technologie dans la fabrication des microprocesseurs pourrait être sa rapidité d'exécution des opérations, malheureusement sa consommation fait que certains constructeurs, comme MOTOROLA ont préféré, pour des utilisations spécifiques rapides, développer un microprocesseur en ***technologie « ECL ».***

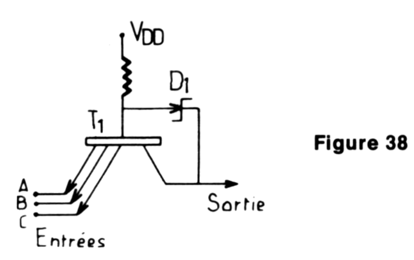


Figure 38

Le schéma de base d'un circuit intégré TTL utilise en entrée un transistor ***« MULTI-EMETTEURS ».*** Dans ce dispositif, le courant à travers la base de T_1 ne se trouve jamais interrompu. Il est constamment aiguillé dans une des deux directions suivantes : soit vers les émetteurs ou soit vers le collecteur de T_1. La charge diffusée dans la base de T_1 est donc toujours disponible au moment même des commutations, ce qui en accroît sensiblement la rapidité. Cependant, quand le transistor conduit, travaillant à saturation et qu'il doit ensuite passer à l'état bloqué, l'intense courant d'écoulement qui traverse l'espace collecteur-base de T_1 avant blocage, ralentit au moment de son passage sa durée de temps de commutation.

On dispose alors une ***diode Schottky*** entre base et collecteur de T_1. La diode Schottky présentant ***un faible direct*** et un ***fort inverse,*** placée ainsi en shunt permet une augmentation de la vitesse de commutation de T_1.

Caractéristiques et emploi des transistors à *Effet de champ* (Voir Fig. n° 39).

Les transistors MOS sont des transistors à effet de champ du type à ***enrichissement*** (enhancement mode), ce qui pour un canal N, se traduit par le fait qu'à $V_{GS} = 0 \rightarrow Idss = 0$ et lorsque V_{GS} devient de plus en plus positif Idss croît (courbe B.)

Les transistors effet de champ ***à jonction*** (***F.E.T.*** = Field Effect Transistor) n'ont pas comme les transistors MOS, la ***GATE isolée,*** ils sont du type à ***appauvrissement*** (dépletion mode) ce qui se traduit pour un canal N par le fait que Idss est maximum pour $V_{GS} = 0$ et que Idss diminue au fur et à mesure que V_{GS} devient négatif (Courbe A).

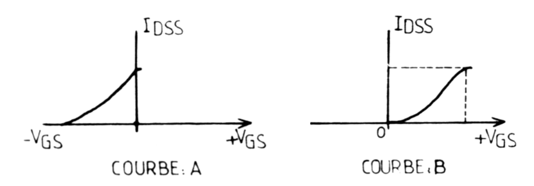


Figure 39

Dans ces 2 types de transistors, on peut fabriquer des circuits pouvant servir au système d'organisation d'un microprocesseur. Ainsi MOTOROLA introduit un PIA, le MC6820A (mode à appauvrissement) alors que son PIA classique MC6820 (MOS canal N) est un Mode à enrichissement.

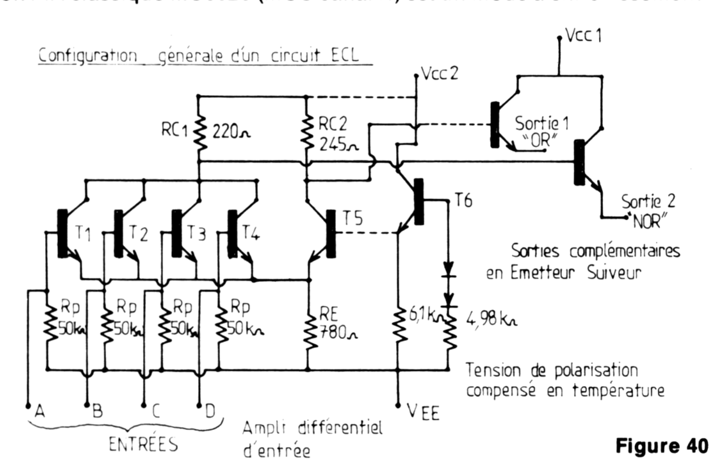


Figure 40

Configuration générale d'un circuit ECL

Nous avons représenté en figure 40 à titre d'exemple une logique ECL. Il s'agit ici de 2 portes complémentaires ***(OR et NOR).***

Cette figure nous montre les entrées (T_1, T_2, T_3, T_4) montées en parallèle et en amplificateur différentiel avec un transistor (T_5) dont la base est

polarisée à l'aide d'une tension de référence prise sur l'Emetteur de T_6. Cette tension de référence est obtenue par un montage classique à compensation de variation de tension en fonction des variations de température.

Il s'agit donc d'une logique à ***Niveaux différentiels non saturés,*** ce qui élimine le temps de déstockage des porteurs dans un transistor lors d'un passage d'un niveau haut à un niveau bas, ou à l'inverse le ***temps de stockage des porteurs*** lors d'un passage d'un niveau bas à un niveau haut.

Cela permet ainsi la production d'une logique très rapide d'où son utilisation dans des systèmes exigeant une ***grande vitesse d'exécution des ordres,*** ce qui peut s'appliquer, d'une manière spécifique à certains microprocesseurs et à leurs mémoires s'insérant dans la composition de leur système de réalisation.

Les autres caractéristiques intéressantes de cette technologie sont :

1. Un haute impédance d'entrée et un gain élevé de tension.

2. Une basse impédance de sortie (émetteur suiveur) d'où la possibilité d'un ***fanout*** important (soit la possibilité d'alimenter en parallèle un grand nombre d'entrées de circuits logiques compatibles).

Technologie I^2L

I^2L signifie (Integration Injection logic) ou logique intégrée à injection (voir Fig. n° 41). C'est une ***logique bipolaire*** qui comporte un circuit d'entrée formé d'un transistor PNP monté base à la masse suivi d'un transistor NPN ***multi-collecteurs.*** Le transistor d'entrée travaille par ***injection de courant*** dans l'émetteur du transistor PNP.

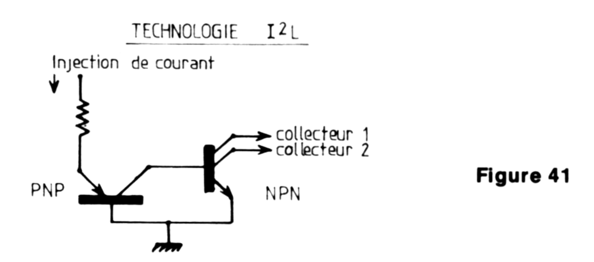


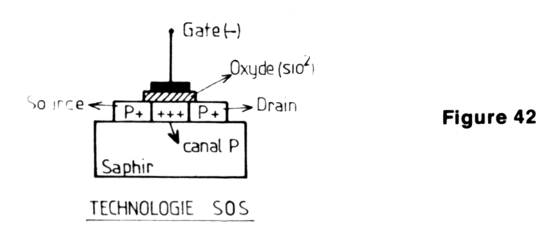
Figure 41

Nous avons déjà sommairement exprimé au début de ce chapitre les qualités de l'I^2L, aussi nous n'en dirons pas plus.

Il existe déjà un microprocesseur fabriqué suivant cette technologie.

Technologie SOS

Afin de terminer ce tour d'horizon sur les technologies actuelles des microprocesseurs et de ses systèmes de mémoires accompagnatrices, il est bon de citer une nouvelle technologie appelée ***SOS*** qui signifie ***Silicon on Sapphire.***



TECHNOLOGIE SOS

Figure 42

La différence existante entre MOS-C-MOS d'une part et SOS d'autre part, est que dans cette dernière technologie on utilise non plus un substrat semi-conducteur de silicium dopé, mais un substrat isolant : Le ***saphir*** comme le montre la figure d'un transistor ***SOS canal P.***

Cette technologie permet au microprocesseur des vitesses d'exécution semblables à celles de la TTL/S sans toutefois consommer plus de puissance qu'en MOS ou en C-MOS.

Afin de terminer ce chapitre sur les technologies de fabrication des microprocesseurs et de leurs systèmes environnants (en particulier les mémoires additionnelles et les circuits d'interface) nous donnons une liste, qui par la suite ne cessera sans doute pas de s'accroître, des microprocesseurs commercialisés et classés suivant leur technologie. Certains de ces microprocesseurs, après avoir été étudiés par un fabricant ont ensuite été adoptés par d'autres qui les réaliseront en seconde source.

Ainsi le microprocesseur ***MOTOROLA MC6800*** a été adopté également en Amérique par ***AMI,*** au Japon par ***HITACHI*** et en Europe par ***EFCIS.***

Microprocesseurs 16 bits et 32 bits VLSI

(Very large scale integration = très haute densité d'intégration)

M68000 et M68008 - Motorola (16 bits d'entrées de données)
8086 - Intel
TMS9900 - Texas
Z8000 - Zilog
M68020 - Motorola - (Mots de 32 bits) en développement

Microprocesseurs - Technologie N-MOS

Mots de 8 bits	M6800 - M6802 - M6805 - M6809 MOTOROLA EA902 - Electronic Arrays F8 - Fairchild 8080 - Intel CMP8 - National S/C 2650 - Signetics TMS8080 - Texas CP1611 - Western Z80 - Zilog
Mots de 16 bits	CP1600 - General instrument
Mots de 12 bits	TLCS12 - Toshiba

Microprocesseurs - Technologie - P-MOS

Mots de 4 bits	PPS25 - Fairchild MCS4004 - Intel MCS040 - Intel IMP4 - National S.C. PPS4 - Rockwell TMS1000 - Texas
Mots de 8 bits	MC58/8008 - Intel 8008/1 - Intel 5065 - MOSTEK IMP8 National SC PPS8 - Rockwell
Mots de 16 bits	PACE - National SC

Microprocesseurs - Technologie C-MOS

Mots de données de 1 bit	MC14500B - MOTOROLA
Mots de 8 bits	COSMAC - RCA et MC146805 - MOTOROLA
Mots de 12	IM100 - Intersil

Microprocesseurs - Technologie ECL

Mots de 4 bits	M10800 - MOTOROLA

Microprocesseurs - Technologie TTL - bipolaire

Par tranches de mots de 4 bits - AM2901 - AMD
3000 - Intel

Microprocesseurs - Technologie I²L

Mots de 4 bits - SB4040 - TEXAS
PP8 - RTC

Cette liste n'est pas exhaustive, on pourrait par exemple y ajouter un microprocesseur un peu particulier, le **MC 6801** de MOTOROLA.

Ce microprocesseur a pour particularité de contenir sur une puce monolithique tout un ensemble de fonctions et de mémoires qui vont en faire un véritable micro-ordinateur (voir chapitre 9).

Enfin, le microprocesseur **MC14500B** comportant des mots de données de 1 bit ***(monobit)*** développe ses instructions sur des mots de 4 bits. Il est extrêmement bon marché. Il ne peut servir utilement à effectuer des calculs, mais il est très rentable pour la prise de décisions (il est décrit en détails dans le livre : « Le microprocesseur en action ») ; c'est ce que l'on peut appeler : « Un Automate programmable ».

CHAPITRE **8**

Organisation du microprocesseur

Nous atteignons ici le cœur d'un micro-ordinateur, c'est-à-dire le ***microprocesseur*** dans lequel les informations arithmétiques ou logiques sont traitées à la demande d'un programmeur, suivant des séquences d'instructions adéquates ; le jeu d'instruction étant spécifique à chaque ensemble personnalisé de micro-ordinateur.

Nous allons expliquer dans ce chapitre, l'organisation générale d'un microprocesseur (schéma synoptique de la figure n° 43) et afin de mieux comprendre sa constitution et son fonctionnement nous ferons ensuite, à titre d'exemple, la présentation d'un microprocesseur bien connu sur le marché, le MC6800 de Motorola.

Tous les microprocesseurs comportent généralement :

1. Un ***Compteur de Programme (PC)*** (Program Counter) appelé encore en français un compteur ordinal

2. Un ***Registre d'Instructions (IR)*** (Instruction Register)

3. Un ***Accumulateur (ACC)***

4. L'***Unité Arithmétique et Logique (ALU)***

5. Un ***Registre d'Etats (CCR)*** (Condition Code Register) appelé encore parfois : « Registre Status »

6. Des ***faisceaux de lignes (BUS)*** reliant le microprocesseur aux autres éléments du système (mémoires et circuits d'interface)

7. Un ensemble constitué d'une ***unité de contrôle*** et un ***décodeur d'instructions (IS).*** « Instruction Decoder ».

1. Le compteur de programme (PC)

C'est un simple registre dont l'information est celle d'un mot binaire de n bits (par exemple : 8 bits). Le PC est chargé de déclencher, séquence

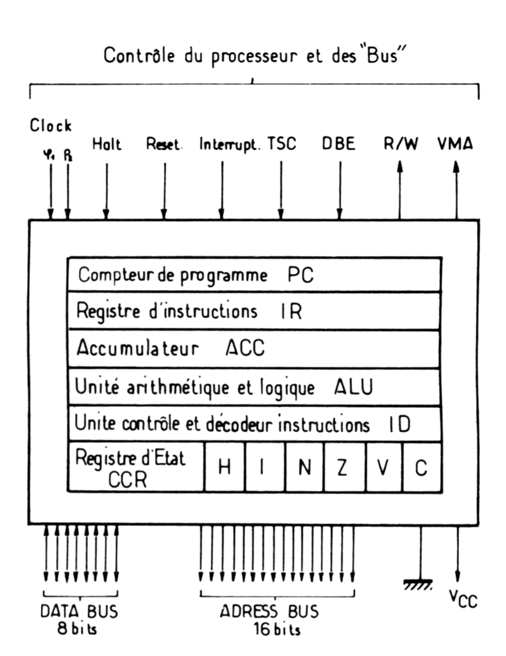


Fig. 43. *Ce schéma représente l'organisation d'un microprocesseur classique (organisation interne + communications externes).*
Nous avons figuré ici un microprocesseur de mots de « 8 bits ».
L'Adress Bus *comporte 16 bits d'adresse, ce qui permet une capacité d'adressage dans la mémoire interne du microprocesseur de 64 K Bytes (2^{16} mots de 8 bits).*

par séquence, le déroulement dans son ordre chronologique des instructions d'un programme à exécuter en désignant au fur et à mesure les adresses où dans la mémoire du micro-ordinateur les instructions sont contenues.

Le PC opère de la façon suivante :

Il ***pointe*** d'abord sur l'adresse de la 1re instruction du programme à effectuer.

Il est ensuite ***Incrémenté de 1*** (c'est-à-dire que l'on ajoute une unité binaire à son ***Mot*** binaire précédent) pendant que s'opère l'exécution de la 1re instruction. Une fois celle-ci accomplie le PC pointe alors sur

l'adresse de la 2e Instruction du programme à effectuer et ensuite s'incrémente à nouveau de « 1 unité binaire ». Les cycles du programme à réaliser se poursuivent ainsi jusqu'à leur achèvement.

Cependant dans certains cas, afin de simplifier la résolution d'un programme on peut avoir à exécuter des instructions déjà stockées dans une autre portion de programme que celle en cours. Le programme modifie alors lui-même le contenu du PC par l'intermédiaire d'instructions spéciales que l'on désigne sous le nom de ***branchements.***

2. Le Registre d'instructions (IR)

L'instruction une fois lue en mémoire doit être ***chargée*** pendant toute la durée de son exécution, dans le ***registre d'instructions.*** Cette instruction comporte généralement deux parties distinctes :

a) Le ***code opération*** qui indique la nature de l'opération à effectuer (ex : ***additionner, charger, incrémenter, brancher...*** etc.)

b) Une ***adresse relative*** à ***l'opérande,*** c'est-à-dire à la donnée visée, ceci afin de pouvoir en disposer en allant la chercher où elle est conservée.

3. L'accumulateur (ACC)

L'accumulateur est un registre qui effectue un travail complémentaire à celui du registre d'instructions. En effet, il sert à charger les données sur lesquelles va porter l'instruction déjà stockée dans le registre d'instructions.

4. L'unité arithmétique et logique (ALU)

C'est l'unité chargée de résoudre les opérations arithmétiques et logiques. Elle comporte au moins 2 entrées :

— Le registre d'instructions doit être branché à la 1re entrée (il transmet à l'ALU l'instruction à réaliser).

— L'accumulateur doit être branché à la 2e entrée. (Il transmet à l'ALU les données sur lesquelles l'instruction devra porter).

A l'intérieur même de l'ALU on dispose de registres devant assurer les opérations arithmétiques et logiques

Suivant le cas, les ***ordres de commande*** serviront à ***activer*** ou non ces différents registres.

Enfin l'ALU doit au moins comporter une sortie permettant de recueillir les résultats des opérations effectuées.

A noter que cette sortie, suivant les besoins, peut être branchée, via le ***Data Bus*** sur une mémoire extérieure au microprocesseur, mais faisant partie de son système micro-ordinateur ou bien encore sur un ***circuit d'interface*** (***I/O*** ou ***PIA***) situé entre le microprocesseur et un périphérique.

Enfin, cette sortie peut également à nouveau être branchée, si cela est nécessaire, sur l'accumulateur.

5. Le Registre d'Etat (CCR)

Comme son nom l'indique, ce registre permet de ***repérer l'Etat*** c'est-à-dire d'indiquer la contenance (niveau logique), à un moment précis d'une partie bien définie du microprocesseur.

Ce registre comporte un certain nombre de ***flip-flop*** ayant chacun une signification particulière.

Les Américains appellent ces flip-flop des ***flags*** c'est-à-dire des ***drapeaux*** pris dans le sens d'***indicateurs***.

Passons en revue les principaux indicateurs existant généralement dans un microprocesseur et voyons comment ils peuvent servir.

a) ***Le carry flip-flop « C »*** encore appelé ***registre de lien « L »*** (traduit de Link) (voir Fig. n° 44).

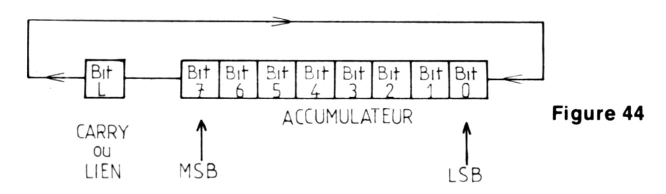


Figure 44

Le flip-flop Carry « C » ou Lien « L » situe son intérêt au niveau de l'accumulateur.

Lors d'une ***opération de rotation,*** il est utilisé afin de servir de ***lien*** pour connecter à travers une boucle le ***MSB*** et le ***LSB*** d'un nombre contenu dans l'accumulateur.

Il peut être encore également considéré dans l'exemple suivant figure n° 45 comme un ***bit d'extension dans l'accumulateur,*** lorsque l'addition de 2 nombres de 8 bits donne lieu à un report de bit « 1 ». C'est en somme un 9e bit de retenue qui ainsi ne sera pas perdu et pris en considération dans une addition portant sur des nombres de 16 bits effectués en deux temps dans un accumulateur 8 bits. Le 9e bit de A conservé dans le registre de lien « L » peut alors être additionné au LSB de B afin de fournir

le résultat correct de l'addition de deux nombres C et D de 16 bits obtenu dans un micro-ordinateur 8 bits travaillant ici en double précision.

Addition simple A dans un micro-ordinateur 8 bits	Nombre de 16 bits	Addition de nombres de 16 bits travaillant à partir d'un système 8 bits en double précision	
10010101 +10011010	C + D	B 01010000 + 10100110	A 10010101 + 10011010
= [1] 00101111 9ᵉ bit de retenue		11110110 + [1] ←	00101111
	= E	= 11110111 MSB LSB	00101111 MSB LSB

Tableau de la figure n° 45

b) ***L'indicateur négatif « N ».*** Flip-flop mis à « 1 » lorsque le contenu de l'accumulateur devient négatif ***(bit de signe)*** voir à ce sujet le chapitre concernant le calcul binaire.

c) ***L'indicateur zéro (z)*** est normalement à 0. Il passe à « 1 » chaque fois que le contenu de l'ACC devient nul (c'est-à-dire lorsqu'il contient 8 bits « 0 »).

Exemple d'applications concernant le bit (C) ou (L) et le Bit (z).

Soit à additionner les nombres binaires de 8 bits :

Primo : Lu en mémoire → 1 1 0 0 1 0 1 1

Secondo Lu dans l'ACC → 0 0 1 1 0 1 0 1

Résultat de l'addition : [1] 0 0 0 0 0 0 0 0

Si l'on examine ce cas d'addition :

Le bit (C) ou (L) devient égal à [1] → indiquant une retenue ou 9ᵉ bit

Le bit (Z) repère en passant à [1] que l'ACC devient nul.

d) ***L'indicateur Overflow (O)*** ou ***(OVF)*** est mis à « 1 » lorsqu'une opération donne lieu à un débordement. C'est le ***Bit de dépassement.***

e) ***L'indicateur Interrupt (I).*** Il permet d'accorder ou de refuser temporairement une demande d'interruption.

f) ***L'indicateur Half-carry (H).*** Il n'est pas utilisé par tous les microprocesseurs. Il est notamment employé dans le MC6800 de Motorola. Il permet d'effectuer une addition arithmétique de 2 nombres posés dans le

système BCD sans pour autant les convertir un à un en binaire naturel, ce qui en simplifie les séquences d'instructions. (Voir à ce sujet l'exemple de la figure n° 46).

Tableau de la figure n° 46

Nombre	Addition de deux nombres décimaux	Position dans le micro	transposition des 2 nombres en système BCD
A	15	ACC	0 0 0 1.0 1 0 1
+ B	+ 28	Mémoire	+ 0 0 1 0.1 0 0 0
= C	= 43		0 0 1 1 1 1 0 1

Résultat de l'addition effectuée par l'ALU en binaire mais est incorrect dans le cas d'une addition en BCD
Si maintenant grâce à une instruction spéciale on ajoute simplement le chiffre + 6 en décimal codé en binaire soit : 0 1 1 0
on obtient :

A + B		0 0 1 1.1 1 0 1
(Additionner en binaire)	+ 6 →	+ 0 0 0 0.0 1 1 0
= C	= 43 →	0 1 0 0.0 0 1 1

Cette fois le chiffre en sytème BCD → 0 1 0 0.0 0 1 1 correspond bien en décimal au nombre 43.

La correction est réalisée dans le microprocesseur MOTOROLA au moyen d'une instruction particulière ***(DAA)*** mais elle engendre également un report sur le bit 4. C'est l'existence de ce report qui affecte précisément l'indicateur ***(H)***. Celui-ci est alors positionné à « 1 ».

Le registre d'état, contenant tous les indicateurs que nous venons d'étudier peut se représenter schématiquement suivant la figure n° 47.

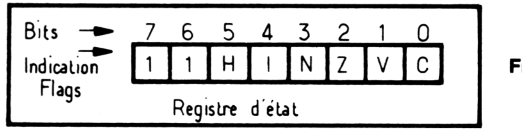


Figure 47

6. Les faisceaux de lignes « BUS »

Nous en avons déjà parlé. Se reporter à ce sujet au chapitre 2, le cerveau humain et l'ordinateur, cerveau robot. Rappelons donc sommairement que nous rencontrons généralement ces « bus » dans 4 utilisations différentes.

— Les ***DATA BUS*** ou bus de données (liaisons bilatérales)

— Les ***Adress BUS*** ou bus d'adresses mémoires (liaisons unilatérales).

— Les ***Control Bus*** ou bus liant l'unité de contrôle du microprocesseur aux autres éléments du micro-ordinateur.

— Les ***Processor Control Bus*** ou bus portant les signaux de commande et contrôle du microprocesseur.

Nous détaillerons tous ces faisceaux de lignes lors de l'étude d'application du microprocesseur MC6800.

7. Unité de contrôle et décodeur d'instructions (ID)

La commande de certaines opérations de fonctionnement effectuées par le microprocesseur est due à des signaux fournis à son unité de contrôle ou de décodage d'instructions.

Le microprocesseur peut également à l'inverse, par l'intermédiaire de son unité de commande, adresser aux autres éléments du micro-ordinateur (mémoires, circuits d'interface) des signaux de déclenchement d'opérations.

Certains s'appliquent uniquement à un microprocesseur défini, d'autres par contre existent pratiquement sur tous ou presque tous les microprocesseurs.

Ce sont ces derniers que nous allons commencer par examiner.

a) ***La Commande d'horloge*** (commande de ***clock***). Cette commande assure un ***cadencement synchrone*** de l'exécution des diverses séquences d'instructions. La précision de ces horloges est généralement définie par la fréquence des signaux d'un oscillateur électronique piloté à l'aide d'un quartz. Les signaux d'horloge peuvent être ***déphasés*** comme le montre la figure n° 48 représentant des signaux à deux phases ***« Q_1 et Q_2 » (Signaux biphasés).*** Ces signaux Q_1 et Q_2 assureront le cadencement de fonctionnement du microprocesseur. On pourrait concevoir un ***cadencement asynchrone*** des microprocesseurs, les instructions étant exécutées à la suite les unes des autres quel que soit le temps nécessaire à la réalisation de chacune d'entre elles. Toutefois, il est à noter que cela oblige à des problèmes de synchronisation avec les éléments externes au microprocesseur parfois bien difficiles à résoudre. Cela est vraiment

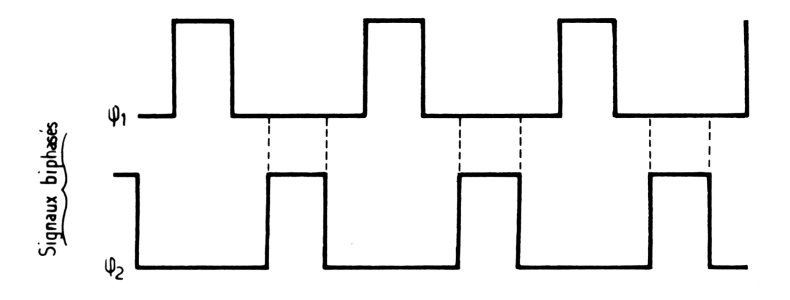


Figure 48 **Cycle d'horloge à 2 phases : Q_1 et Q_2**

dommage, car l'exécution d'un cadencement asynchrone des instructions est évidemment plus rapide que celle d'un cadencement synchrone qui doit obligatoirement compter des temps morts entre chaque séquence d'instructions.

b) ***Le signal d'arrêt d'un programme (Halt)*** ou bien encore ***(stop).*** Il est produit, soit à la fin d'un programme, soit à la suite d'une demande d'interruption de programme.

c) ***Le signal de remise en route d'instructions. (Reset)*** Ce signal est utilisé pour la mise en route du microprocesseur ou bien encore pour sa remise en route à la suite par exemple d'une panne d'alimentation générale. D'autres noms peuvent être attribués tel que : ***START*** pour la mise en route et ***RESTART*** pour la remise en route.

d) ***Le signal de demande d'interruption de programme (IRQ).*** (Interrupt Request). Ce signal peut être émis lorsqu'un périphérique fait au microprocesseur une demande urgente.

Le microprocesseur arrêtera son programme en cours afin de satisfaire cette demande extérieure, après quoi, il pourra reprendre et continuer le 1er programme à la séquence suivant celle où il s'était précédemment arrêté.

e) ***Le signal de commande de 3e état.*** « Three State Control ». ***(TSC).*** Nous en avons déjà parlé au cours du 2e chapitre. Prière de bien vouloir vous y reporter.

f) ***Le signal de fonctionnement du Data Bus*** « Data Bus enable » (D.B.E.). Ce signal établit la liaison entre l'unité de contrôle et de commande du microprocesseur et les différentes mémoires du système où sont stockées des données.

Nous avons examiné les principaux signaux pouvant agir sur l'unité de contrôle d'un microprocesseur.

Il nous reste à voir maintenant les principaux signaux issus de cette unité de contrôle et commandant les circuits du micro-ordinateur externes au microprocesseur (mémoires et circuits d'interface avec périphériques).

a) ***Le Signal Read-Write (R/W)***. C'est le signal permettant de lire ou d'écrire une donnée en mémoire, la liaison entre le microprocesseur et cette mémoire étant assure à l'aide du « data bus ». Par exemple, supposons que pour un niveau logique $\mathscr{H}$ nous soyons dans la position ***Read*** (lecture). En ce cas, nous lirons la donnée en mémoire, qui circulera dans la « Data Bus » dans le sens mémoire → microprocesseur afin de charger cette donnée dans l'accumulateur du microprocesseur ; par contre sur un niveau logique $\mathscr{B}$, nous serons alors dans la position ***Write*** c'est-à-dire que nous inscrirons en mémoire la donnée qui vient d'être élaborée dans ***l'ALU*** du microprocesseur, les signaux dans le « Data Bus » circulant alors dans le sens : microprocesseur → mémoire.

b) ***Le signal de validation : Mémoire → Adress Bus (VMA).*** Ce signal valide une adresse sur ***l'Adress Bus*** qui peut ***activer*** une RAM ou bien encore des interfaces avec les périphériques tel qu'un « PIA ».

8. Le Microprocesseur MC6800

Maintenant que nous avons défini sommairement l'organisation générale d'un microprocesseur, nous allons décrire, pour nous familiariser avec ceux-ci, l'un d'entre eux existant sur le marché : Le ***MC 6800 de MOTOROLA.*** Cette étude nous permettra surtout d'entrevoir comment l'on peut s'en servir.

L'exposé va comporter les paragraphes suivants :

— 8.1 - Généralités
— 8.2 - Brochage
— 8.3 - Configuration synoptique
— 8.4 - Points particuliers au MC 6800
— 8.5 - Jeu d'instructions - Le langage mnémonique
— 8.6 - ***Modes d'adressage***
— 8.7 - Tableaux de manipulation des instructions
    a) Dans l'accumulateur et la mémoire
    b) Dans le ***Registre d'index*** et le stack
    c) Instructions de ***Sauts*** et de ***branchements***
    d) Dans le ***registre status***

8.1 - Généralités - Le microprocesseur MC6800 de Motorola est un microprocesseur monolithique 8 bits. Technologie MOS canal N.

Gate silicium. Il est encapsulé dans un boîtier DIL comportant 40 broches de sorties.

Le MC6800 est l'unité centrale, qui associée aux autres circuits intégrés (éléments mémoires ou d'interfaces avec les périphériques) de la famille M6800 déterminent des systèmes micro-ordinateurs. Le choix et la composition des circuits de la famille M6800 peuvent être réalisés en fonction des besoins d'utilisation spécifique.

Les produits de la famille M6800, dont le MC6800 sont compatibles avec les produits de la famille TTL. Ils sont alimentés à l'aide d'une seule tension de + 5 Volts. Il n'y a pas besoin de « Buffers externes TTL » en liaison avec les circuits BUS d'interface.

Le MC6800 doté d'un « Adress Bus » 16 bits a une capacité d'adressage mémoire de 65 K bytes. L'adressage en mémoire peut s'effectuer suivant 7 **MODES** différents. Nous expliquons plus loin comment s'organise chacun de ces modes.

Voici la désignation de ceux-ci en français et en américain :

Modes d'adressages : Accumulateur - accumulateur (ACCX) adressing.
- ***Immédiat - Immediate adressing***
- ***Direct - Direct Adressing***
- ***Etendu - Extended Adressing***
- ***Indexé - Indexed Adressing***
- ***Impliqué - Implied Adressing***
- ***Relatif - Relative Adressing.***

Le « Data Bus » du MC6800, est bidirectionnel, doté d'une logique à 3 états : niveaux $\mathcal{H}$ et $\mathcal{B}$ et impédance infinie.

Le MC6800 comporte six registres internes :

— Deux accumulateurs : ***A*** et ***B*** + l'***Alu***
— Un ***Registre d'index***

Un registre de sauvegarde des données appelé : ***Stack Pointer,*** un registre d'état, un registre d'instructions et le « program-counter ».

Nous expliquerons le fonctionnement des registres spéciaux au MC6800 c'est-à-dire le Stack Pointer et le registre d'index par la suite.

Les instructions en langage mnémonique sont au nombre de 72.

Le signal d'horloge est biphasé (Q_1 et Q_2) et obtenu à partir d'un quartz 1 MHz

Il peut travailler avec un périphérique en DMA.

8.2 - Brochage du boîtier MC6800
(voir Fig. n° 49)

8.3 - Configuration synoptique du MC6800
(voir Fig. n° 50)

8.4 - Points particuliers au MC6800

Les particularités du MC6800 que nous allons examiner sont les suivantes :

a) Existence de 2 accumulateurs ***(ACCA*** et ***ACCB)*** au lieu d'un seul accumulateur.

b) Existence de 2 registres spéciaux, l'un appelé le ***Stack Pointer*** et l'autre appelé le ***Registre d'index (IR).***

Examen des 2 accumulateurs :

Ils permettent :

a) Une économie en temps d'exécution

b) Une économie en positions mémoires (moins d'instructions et de données à y faire figurer).

En effet on peut d'abord réaliser le stockage d'un résultat partiel programmé dans un accumulateur (par exemple, le nombre 4 dans ACCB) pendant que l'on effectue un calcul dans l'autre accumulateur (par exemple 2 + 5 dans l'accumulateur ACCA).

Enfin on peut additionner 4 à 7 en transférant le contenu de ACCB dans ACCA.

Le calcul ainsi effectué est évidemment moins long que si l'on avait eu à aller rechercher le chiffre 4 dans une mémoire extérieure au CPU, de plus on a de ce fait économisé de la place dans la mémoire.

Examen du « Stack Pointer » : (SP)

Le « Stack Pointer » est une pile LIFO (voir explication au chapitre 2).

Son importance se manifeste lors du traitement des interruptions ou des sub-routines.

On doit en effet sauvegarder en mémoire une partie d'un programme en cours, le temps que se développe un autre programme résultant d'une demande d'interruption prioritaire d'un périphérique ***(NMI)*** ou bien encore le temps que se développe une demande de sous-programme ou ***Sub-Routine*** commandée par une instruction de saut ***(JUMP)***

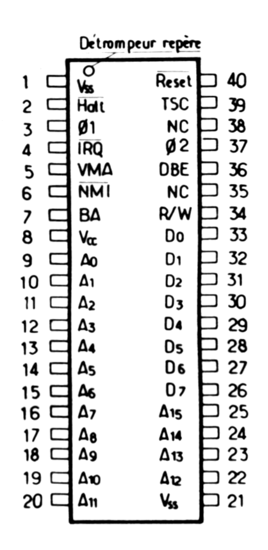


Figure 49

Brochage du MC6800

Tout d'abord lorsque l'instruction est barrée (Exemple : $\overline{RESET}$) cela signifie que la validation est effective pour un signal $\mathcal{B} = 0$ en logique positive. Lorsque l'instruction n'est pas barrée (exemple : TSC), la validation est effective pour un signal $\mathcal{H} = 1$ en logique positive.

A_0 à A_{15} → *lignes du faisceau d'Adress Bus*

D_0 à D_7 → *lignes du faisceau de Data Bus*

Vss = *masse, tension de référence (broches 1 et 21)*

Vcc = *tension d'alimentation positive (broche 8).*

Instructions spéciales au MC6800

BA - Bus disponible (Bus available)

IRQ - *Demande d'interruption. Le programme en cours se termine avant de donner priorité à la demande d'un périphérique.*

NMI - *Interruption non masquée - (non Maskable Interrupt). Le programme en cours est interrompu dès la fin de la séquence d'instructions en cours et priorité est alors donnée à la demande d'un périphérique. Le programme du périphérique étant épuisé, le programme précédent interrompu est repris jusqu'à son achèvement.*

Commentaires sur la figure 50.

Le microprocesseur MC6800 est un 8 bits.

L'adressage se réalisant sur 16 bits afin d'augmenter la capacité mémoire, le micro travaille en double précision c'est-à-dire qu'il adresse sur 2 mots de 8 bits. Le ***PC***, le ***SP*** et l'***IR*** travaillant chacun sur 16 bits d'adressage sont représentés sur le schéma en 2 parties de 8 bits : la partie L (Less = mot de faible poids) et la partie M (Most = mot de poids le plus fort). Par contre l'accumulateur A comme l'accumulateur B travaillent chacun uniquement sur des mots de 8 bits ainsi que le BUS des sorties bilatérales de données (↕). Les sorties de « Bus Adress » unilatérales (↑) sont divisées en 2 parties de 8 bits.

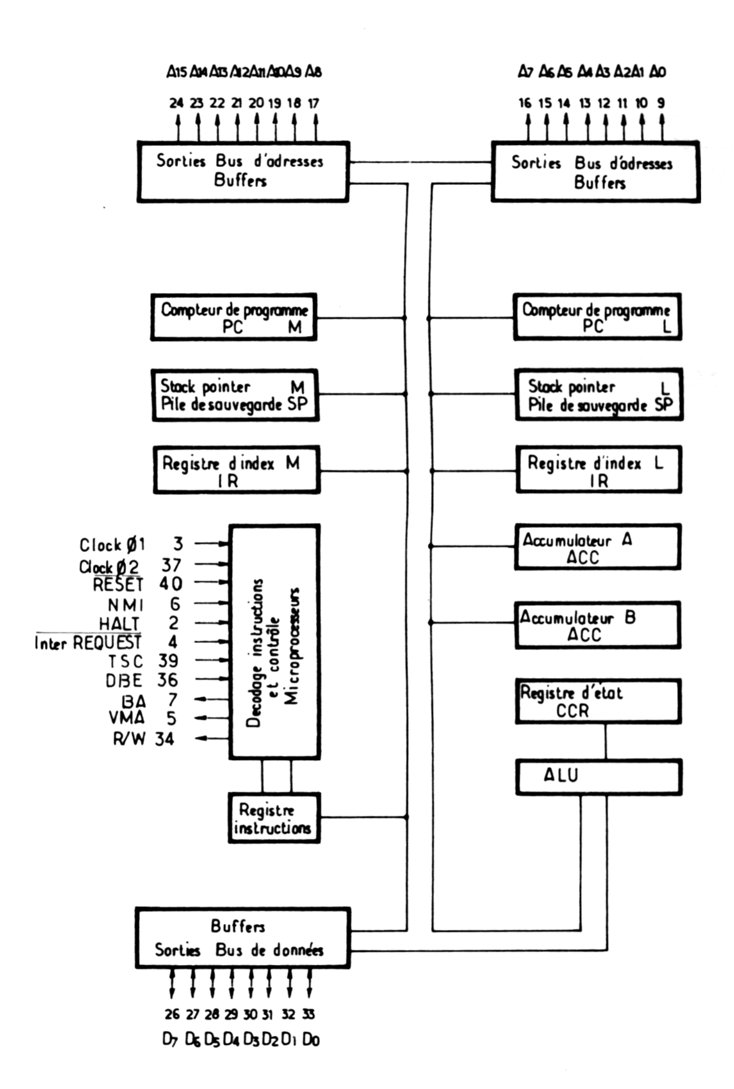


Fig. 50. — *Configuration synoptique du MC6800*

Le traitement d'un Stack Pointer fait intervenir 2 opérations :

L'opération de ***Pushing*** pendant laquelle l'accumulateur est vidé dans la mémoire du Stack Pointer à une adresse définie.

L'opération de ***Popping*** pendant laquelle, à l'inverse, la donnée stockée dans la mémoire du SP est vidée dans l'accumulateur.

Un exemple et une explication du fonctionnement du ***stack*** sont donnés en figure n° 51.

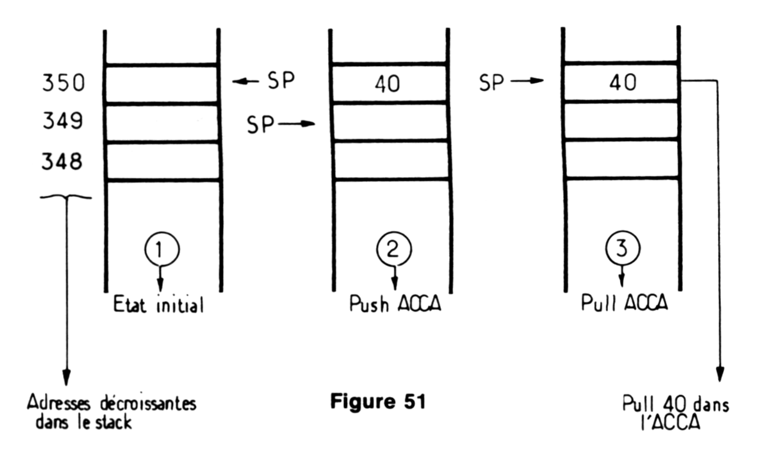


Figure 51

① → Avant de commencer une opération de pushing : ***initialiser*** le SP à l'adresse disponible de départ (exemple : 350)

② → Opération de ***pushing :*** Le contenu de l'ACCA (exemple 40) est chargé ***(LOAD)*** à l'adresse 350 et le SP est ***décrémenté*** de « 1 ». Il pointe donc sur l'adresse 349 (350-1).

③ → Opération de ***popping :*** Le SP est ***incrémenté*** de « 1 ». Il pointe sur l'adresse 350 et vide son contenu (40) dans l'ACCA. (349 + 1).

④ → Dans l'opération de Pushing, le SP étant pointé sur l'adresse 349, on aurait pu également y introduire le contenu éventuel de l'ACCB et ensuite ***décrémenter*** à nouveau de « 1 ». L'opération de pushing étant achevée et les accumulateurs vidés (contenu = 0), le microprocesseur peut effectuer après un ***interrupt,*** la demande programmée par un périphérique ou bien encore après une instruction de ***saut*** (Jump) une demande de ***sub-routine.*** Ces demandes réalisées, on peut alors passer à l'opération de ***popping*** pour récupérer le programme initial en cours et le continuer normalement jusqu'à son achèvement.

Examen du Registre d'index

Il permet d'exécuter les instructions suivant le mode d'adressage indexé. (Nous verrons le fonctionnement de ce mode au paragraphe réservé aux différents modes d'adressage).

8.5 - Le jeu d'instructions du MC6800 et son langage mnémonique

1. ABA - Addition de l'accumulateur B à l'accumulateur A
2. ADC - Addition avec retenue
3. ADD - Addition
4. AND - « ET » logique
5. ASL - Décalage arithmétique de un vers la gauche
6. ASR - Décalage arithmétique de un vers la droite

7. BCC - Branchement s'il n'y a pas de retenue
8. BCS - Branchement s'il y a retenue
9. BEQ - Branchement si égal (à zéro)
10. BGE - Branchement si supérieur ou égal à zéro
11. BGT - Branchement si plus grand que zéro
12. BHI - Branchement si supérieur
13. BIT - Test de bits
14. BLE - Branchement si inférieur ou égal à zéro
15. BLS Branchement si inférieur ou égal
16. BLT - Branchement si inférieur à zéro
17. BMI - Branchement si négatif
18. BNE - Branchement si non nul
19. BPL - Branchement si positif ou nul
20. BRA - Branchement inconditionnel
21. BSR - Branchement à un sous-programme
22. BVC - Branchement si pas de dépassement
23. BVS - Branchement si dépassement

24. CBA - Comparaison des accumulateurs
25. CLC - Mise à zéro du bit de retenue
26. CLI - Mise à zéro du masque d'interruption
27. CLR - Mise à zéro
28. CLV - Mise à zéro du bit de dépassem. en compl. à deux
29. CMP - Comparaison
30. COM - Complément à un
31. CPX - Comparaison du registre d'index

32. DAA - Ajustement décimal sur l'accumulateur A
33. DEC - Décrémentation
34. DES - Décrémentation du pointeur de pile
35. DEX - Décrémentation du registre d'index

36. EOR - « OU » exclusif

37. INC - Incrémentation
38. INS - Incrémentation du pointeur de pile
39. INX - Incrémentation du registre d'index

40. JMP - Saut inconditionnel

41. JSR - Saut à un sous-programme

42. LDA - Chargement accumulateur
43. LDS - Chargement du pointeur de pile
44. LDX - Chargement du registre d'index
45. LSR - Décalage logique vers la droite d'une position

46. NEG - Complément à deux (opposé)
47. NOP - Passage en séquence (non opération)

48. ORA « « OU » logique

49. PSH - Mise d'un octet dans la pile
50. PUL - Extraction d'un octet de la pile

51. ROL - Décalage circulaire à gauche
52. ROR - Décalage circulaire à droite
53. RTI - Retour de séquence d'interruption
54. RTS - Retour de sous-programme

55. SBA - Soustraction entre accumulateurs
56. SBC - Soustraction avec retenue
57. SEC - Mise à un de la retenue
58. SEI - Mise à un du masque d'interruption
59. SEV - Mise à un du bit de dépassement en complément à deux
60. STA - Mise en mémoire d'un accumulateur
61. STS - Mise en mémoire du pointeur de pile
62. STX - Mise en mémoire du registre d'index
63. SUB - Soustraction
64. SWI - Interruption programmée

65. TAB - Transfert de l'accumulateur A dans l'accumulateur B
66. TAP - Transfert de l'accumulateur A dans le registre d'état
67. TBA - Transfert de l'accumulateur B dans l'accumulateur A
68. TPA - Transfert du registre d'état dans l'accumulateur A
69. TST - Test
70. TSX - Transfert du pointeur de pile dans le registre d'index
71. TSX - Transfert du registre d'index dans le pointeur de pile
72. WAI - Attente d'interruption

Nous reconnaissons ici la plupart des instructions que nous avons pu mentionner déjà au cours de l'énoncé des chapitres précédents. Ainsi les instructions 1 à 3, sont des instructions ***d'addition*** et par exemple l'instruction 2 doit tenir compte de l'indicateur ***Carry « C ».***

Les instructions 55, 56 et 63 servent à effectuer des ***soustractions.***

Les instructions 4, 36 et 48 réalisent des opérations de ***fonctions logiques.***

Les instructions 7 à 23 servent à exécuter des ***branchements***.

Les instructions 25 à 28 servent à ***effacer (clear)*** les bits positionnés dans les divers registres mentionnés du microprocesseur.

Les instructions 33 à 35 servent à ***décrémenter*** le registre spécifié.

Les instructions 37 à 39 servent à ***incrémenter*** le registre spécifié.

Les instructions 42 à 44 à ***charger (load)*** le registre spécifié.

Les instructions 60 à 62 vont permettre de ***stocker (store)*** le registre spécifié.

Les instructions 65 à 71 servent à effectuer des ***opérations de transfert*** d'un registre à un autre spécifié... etc.

Une question demeure : comment utiliser d'une manière pratique toutes ces instructions, autrement dit, comment programmer nos besoins afin que le microprocesseur puisse nous renseigner.

Si l'on désire connaître tous les détails concernant la programmation il est nécessaire de consulter le manuel de programmation spécifique à chaque microprocesseur utilisé, ainsi à titre d'exemple le manuel de programmation du MC6800 comporte plus de 150 pages.

Cependant dans le cadre de cet exposé, nous allons maintenant examiner les différents ***Modes d'adressage*** et pour se faire une idée plus précise quant à la programmation, nous reproduirons les tableaux de ***manipulation des instructions.***

8.6 - Les différents modes d'adressage

Nous avons pu voir que le registre d'instructions comporte 2 parties : *a)* L'instruction proprement dite appelée ***code opération (OP)*** écrite en langage mnémonique et *b)* ***L'adresse relative à l'opérande*** qui permet d'aller chercher la donnée visée par l'instruction. Le tableau de la figure nº 52 nous donne pour chaque instruction, ses possibilités d'adressage et le temps d'horloge d'exécution en ***cycles machine.*** Le signal d'horloge étant synchronisé avec un quartz de 1 MHz, on peut dire qu'un cycle machine est égal à 1 microseconde.

Adressage accumulateur (ACCX). Le code opération comporte 3 lettres, l'adresse de l'opérande ***(X)*** est soit ***(A)*** pour désigner l'accumulateur A, soit ***(B)*** pour désigner l'accumulateur B. L'Assembleur traduit l'adressage en code machine ou code objet sur un seul octet (mot de 8 bits). Les instructions adressées dans la colonne ACCX peuvent également être utilisées soit avec le mode ***étendu*** (Extended), soit avec le mode ***indexé.***

Fig. 52. — Tableau des instructions, de leur mode d'adressage et de leur temps d'exécution en cycles-machine.

	Double opérande	ACCX	Immédiat (#)	Direct	Étendu	Indexé	Impliqué	Relatif		Double opérande	ACCX	Immédiat (#)	Direct	Étendu	Indexé	Implique
ABA		•	•	•	•	•	2	•	INC		2	•	•	6	7	•
ADC	x	•	2	3	4	5	•	•	INS		•	•	•	•	•	4
ADD	x	•	2	3	4	5	•	•	INX		•	•	•	•	•	4
AND	x	•	2	3	4	5	•	•	JMP		•	•	•	3	4	•
ASL		2	•	•	6	7	•	•	JSR		•	•	•	9	8	•
ASR		2	•	•	6	7	•	•	LDA	x	•	2	3	4	5	•
BCC		•	•	•	•	•	•	4	LDS		•	3	4	5	6	•
BCS		•	•	•	•	•	•	4	LDX		•	3	4	5	6	•
BEA		•	•	•	•	•	•	4	LSR		2	•	•	6	7	•
BGE		•	•	•	•	•	•	4	NEG		2	•	•	6	7	•
BGT		•	•	•	•	•	•	4	NOP		•	•	•	•	•	2
BHI		•	•	•	•	•	•	4	ORA	x	•	2	3	4	5	•
BIT	x	•	2	3	4	5	•	•	PSH		•	•	•	•	•	4
BLE		•	•	•	•	•	•	4	PUL		•	•	•	•	•	4
BLS		•	•	•	•	•	•	4	ROL		2	•	•	6	7	•
BLT		•	•	•	•	•	•	4	ROR		2	•	•	6	7	•
BMI		•	•	•	•	•	•	4	RTI		•	•	•	•	•	10
BNE		•	•	•	•	•	•	4	RTS		•	•	•	•	•	5
BPL		•	•	•	•	•	•	4	SBA		•	•	•	•	•	2
BRA		•	•	•	•	•	•	4	SBC	x	•	2	3	4	5	•
BSR		•	•	•	•	•	•	8	SEC		•	•	•	•	•	2
BVC		•	•	•	•	•	•	4	SEI		•	•	•	•	•	2
BVS		•	•	•	•	•	•	4	SEV		•	•	•	•	•	2
CBA		•	•	•	•	•	2	•	STA	x	•	•	4	5	6	•
CLC		•	•	•	•	•	2	•	STS		•	•	5	6	7	•
CLI		•	•	•	•	•	2	•	STX		•	•	5	6	7	•
CLR		2	•	•	6	7	•	•	SUB	x	•	2	3	4	5	•
CLV		•	•	•	•	•	2	•	SWI		•	•	•	•	•	12
CMP	x	•	2	3	4	5	•	•	TAB		•	•	•	•	•	2
COM		2	•	•	6	7	•	•	TAP		•	•	•	•	•	2
CPX		•	3	4	5	6	•	•	TBA		•	•	•	•	•	2
DAA		•	•	•	•	•	2	•	TPA		•	•	•	•	•	2
DEC		2	•	•	6	7	•	•	TST		2	•	•	6	7	•
DES		•	•	•	•	•	4	•	TSX		•	•	•	•	•	4
DEX		•	•	•	•	•	4	•	TSX		•	•	•	•	•	4
EOR	x	•	2	3	4	5	•	•	WAI		•	•	•	•	•	9

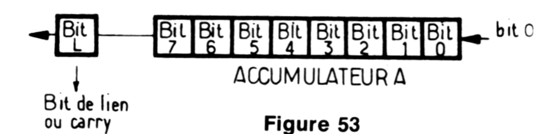


Figure 53

*Exécution instruction « **ASLA** »*
Exemple : introduction de 0 dans le bit 0 et décalage du nombre binaire vers la gauche d'un bit.
— Le bit 0 prend la place du bit 1
— Le bit 1 prend la place du bit 2... etc.
et le bit 7 est conservé dans le bit « L ».

Exemple de fonctionnement : (voir Fig. n° 53)

Soit ***ASLA*** (Instruction de ***décalage vers la gauche)*** des bits d'un nombre arithmétique dans l'accumulateur A.

L'instruction source ASLA est traduite en langage machine par un mot binaire de 8 bits (1 octet) et l'exécution comprend au total 2 cycles machine.

Pendant le 1er cycle, seulement « l'address Bus » et le « data Bus » sont activés. La donnée en mémoire est chargée dans l'accumulateur A.

Pendant le 2e cycle, la machine exécute l'opération demandée par l'instruction et ***« VMA »*** est activé (niveau ℋ).

Adressage immédiat (#). Le signe indiquant ce mode d'adressage en langage source, après le code opération est : #. Les instructions traduites en langage objet comportent 2 ou 3 octets.

Dans cet adressage, l'opérande n'est pas inscrit en mémoire, c'est le programmeur qui, par exemple, le frappe sur le clavier de l'imprimante de son périphérique, l'adressant ainsi au microprocesseur via ses interfaces.

L'opérande est contenu dans le 2e octet excepté pour les instructions ***LDS, LDX*** et ***CPX*** qui demandent à ce que l'opérande figure dans le second et le troisième octet.

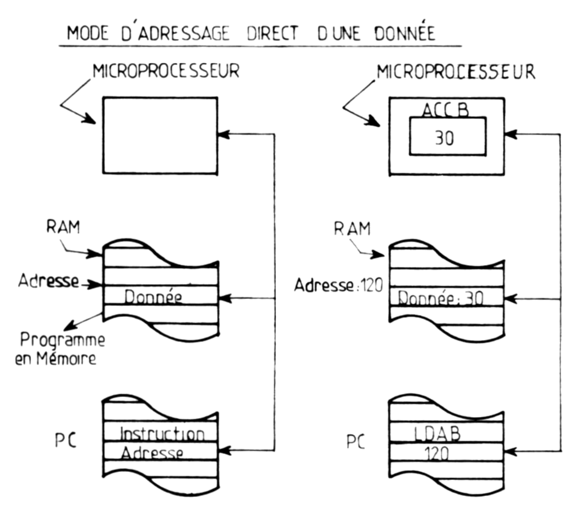
Adressage direct ou adressage absolu - Les instructions traduites en code machine comportent 2 octets. Dans cet adressage, l'instruction contient l'***adresse effective*** de la mémoire. L'adresse est située dans le second octet de l'instruction. L'adressage direct permet d'aller chercher directement les 256 mots (2^8 possibilités) les plus bas dans la mémoire,

c'est-à-dire ceux qui sont contenus dans les adresses 0 à 255. Dans la plupart des configurations de micro-ordinateur, ces données du programme sont emmagasinées dans une ***RAM.***

Adressage étendu (extended). Comme dans l'adressage direct, l'instruction contient l'adresse effective de la mémoire. Les instructions tra-

Figure 54

Pour une bonne compréhension des modes d'adressage du MC6800 nous donnons ici un schéma explicatif du mode d'adressage direct d'une donnée.



L'adressage se fait sur un octet. Il est donc compris entre l'adresse 0 et l'adresse 255.

Le PC pointe sur l'adresse : 120. La donnée à cette adresse dans la RAM est : 30 (NB traduit en code binaire c.a.d. en langage objet). La donnée est introduite dans l'accumulateur B.

duites en code machine comportent 3 octets, le 3e octet de l'instruction détermine le ***mot de plus faible poids de l'adresse de l'opérande,*** le 2e octet étant le ***mot de plus fort poids*** quant à cette adresse.

L'assembleur sélectionne le mode d'adressage étendu chaque fois que l'adresse est supérieure à la position 255. Quand l'adresse dans la mémoire est inférieure à 256, l'assembleur sélectionne le mode d'adressage direct.

Adressage indexé (Indexed). Il comporte une instruction en langage machine de 2 octets. Il s'obtient grâce à un registre spécial appelé : ***registre d'index.*** Le caractère ***(X)*** étant employé pour désigner le registre d'index, l'assembleur sélectionne le ***mode d'adressage indexé*** chaque fois que ce caractère lui est transmis dans le code opération (OP).

Exemple : ***INX*** → signifie incrémenter le registre d'index.
LDX → charger le registre d'index.
STX → emmagasiner dans le registre d'index... etc.

Le registre d'index est un registre ***16 bits*** qui facilite l'accès à des données séquentielles stockées en mémoire. Dans le mode d'adressage indexé, on obtient l'adresse effective en mémoire de l'opérande en faisant la somme : ***B + X, B*** étant l'adresse fournie par l'instruction et ***X***, le contenu du registre d'index. Le résultat conservé temporairement pendant l'exécution de la séquence d'instruction, dans un ***registre d'adresse (AR)*** = « Adress Register » définit l'adresse effective choisie dans la mémoire.

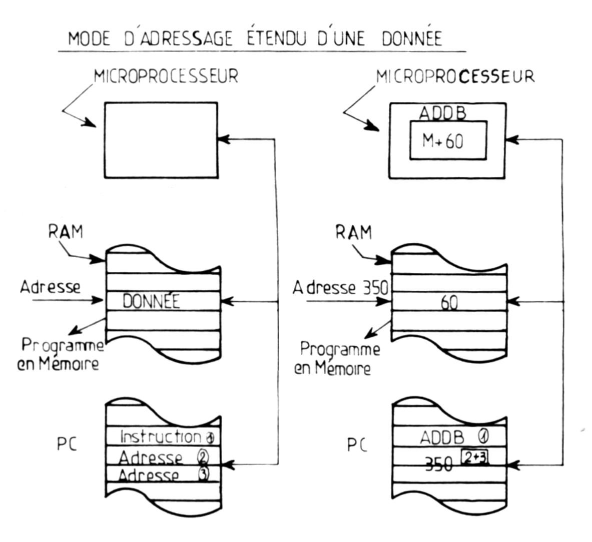
Après chaque séquence le registre d'index est ***incrémenté,*** de ce fait tous les mots d'une table complète peuvent être adressés à partir de la même instruction, si ceux-ci ont été programmés en mémoire dans un ordre chronologique croissant d'adressage.

Adressage impliqué (implied) ou ***inhérent.*** L'instruction se traduit en code machine sur un octet. Dans ce mode d'adressage impliqué, le code opération de l'instruction (OP), suffit à lui-même pour indiquer l'adresse de l'opérande. Par exemple : le code opération ***ABA*** (addition dans les accumulateurs) demande deux opérandes qui sont localisées dans l'accumulateur A et dans l'accumulateur B du microprocesseur. Le code opération détermine également que le résultat de l'exécution de l'instruction sera gardé dans l'accumulateur A.

Il existe ***25 instructions*** dont le code opération indique lui-même l'adresse, c'est-à-dire le registre dans lequel se trouve l'opérande (accumulateur, registre d'Etat, stack pointer... etc.).

Figure 55

Pour une bonne compréhension des modes d'adressage du MC6800, nous donnons ici un schéma explicatif du mode d'adressage étendu d'une donnée.



L'adressage se fait sur 2 octets ② et ③.
L'instruction se traduit avec l'octet ①. Il supplée à l'adressage direct chaque fois que l'adresse est égale ou supérieure à la 256ᵉ dans la mémoire RAM.

M contenu dans l'accumulateur avec un nombre contenu dans l'adresse 350 de la mémoire RAM (nombre 60).
PC pointe sur l'adresse 350 de la RAM (liaison par adress Bus).
Le Nb 60 est lu dans la RAM et vient charger l'accumulateur B. (liaison RAM → MPU par la data bus).
L'instruction d'additionner 60 avec le Nb M s'exécute ensuite dans l'ALU.

Attention... ! certains codes opérations d'une instruction comme ***LDS, LDX*** définissent également l'adresse de destination-registre, cependant l'instruction comporte un autre mode d'adressage, car l'opérande est d'abord à aller chercher dans une autre mémoire que celle du registre de destination. Le code en langage machine de l'instruction se lit alors non plus sur 1 octet mais sur 2 ou 3 octets, 1 octet étant toujours réservé au code opération de l'instruction, le ou les autres octets étant réservés à déterminer l'adresse de l'opérande.

Adressage relatif - L'instruction de ce mode d'adressage relatif comporte 2 octets. Le mode d'adressage relatif s'applique ici aux opérations de ***branchements (branch).*** On sait que ces opérations permettent par exemple l'éxécution d'instructions de sub-routines ***(BSR)*** déjà stockées dans un certain nombre d'adresses situées dans une autre portion de mémoire que celle où se déroule le programme en cours. Il va donc falloir modifier le pointage successif d'adresses du programme en cours dans le PC (compteur de programme). Juste avant de commencer cette modification, le PC pointait sur une adresse du programme en cours. Cette adresse va s'appeler ***adresse de référence.*** Elle va permettre de déterminer l'***Adresse effective*** du nouveau programme en l'ajoutant à l'adresse relative de l'instruction de « Branch ». Cette opération s'opère de la façon suivante :

L'adresse contenue dans le 2[e] Octet de l'instruction de « branch » est additionnée avec celle de l'octet de plus faible poids du compteur de programme, puis on incrémente 2 fois le PC de « 1 ». Le contenu du carry « C » est alors additionné à l'octet de poids le plus fort du PC.

La règle qui s'applique au mode d'adressage relatif est que l'adresse de destination de l'instruction de « branch » (adresse effective) doit être comprise à l'intérieur d'une gamme de — 126 à + 129 octets de l'instruction présente soit :

$$(PC + 2) - 128 \leq D \leq (PC + 2) + 127$$

Le mode d'adressage relatif ne s'appliquant qu'aux seules opérations de « Branch », le code opération de 3 lettres de l'instruction suffit à l'assembleur lui-même, sans qu'il soit nécessaire au programmeur d'indiquer ce mode d'adressage par un signe quelconque.

Double opérande ou ***double adressage d'opérande.***

11 instructions ont une possibilité de double adressage (voir croix du tableau de la figure n° 52) (voir également tableaux n° 56 et 57). Le 1[er] adressage correspond à l'accumulateur A, le 2[e] adressage à l'accumulateur B. Le caractère qui le désigne A ou B est écrit derrière les 3 caractères du code opération de l'instruction.

8.7 - Tableaux de manipulation des instructions

Pour terminer ce chapitre sur le microprocesseur, nous donnons à la suite des modes d'adressage, les tableaux de manipulation des instructions dans les divers registres du microprocesseur MC6800 avec la légende propre à ces tableaux.

En ce qui concerne les chiffres (1) à (13) entourés dans les « Flip-Flop » du registre d'état, de plus amples détails sont parfois nécessaires pour une bonne compréhension. Nous conseillons à nos lecteurs de bien vouloir se reporter au livre de programmation manuelle du MC6800 édité par Motorola.

— A toutes fins utiles signalons que les termes sous la colonne « Opérations » des tableaux des figures 56, 57, 59 et 60 et sous la colonne « Pointer opérations » du tableau de la figure 58 sont explicités en français au paragraphe 8.5 du jeu d'instructions du MC6800 pages 85 et 86.

Vous pouvez donc vous y reporter pour une meilleure compréhension de ceux-ci.

Fig. 56 - 1er tableau des instructions accumulateurs et mémoires

		Modes d'adressage															Opérations logiques et arithmétiques	Registres d'état					
		IMMED			DIRECT			INDEX			EXTND			IMPLIED				5	4	3	2	1	0
OPERATIONS	MNEMONIC	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A · M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B · M → B	•	•	↕	↕	R	•
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A · M	•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B · M	•	•	↕	↕	R	•
Clear	CLR							6F	7	2	7F	6	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A − M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B − M	•	•	↕	↕	↕	↕
Compare Acmltrs	CBA													11	2	1	A − B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				$\overline{M}$ → M	•	•	↕	↕	R	S
	COMA													43	2	1	$\overline{A}$ → A	•	•	↕	↕	R	S
	COMB													53	2	1	$\overline{B}$ → B	•	•	↕	↕	R	S
Complement, 2's	NEG							60	7	2	70	6	3				00 − M → M	•	•	↕	↕	①	②
(Negate)	NEGA													40	2	1	00 − A → A	•	•	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts Binary Add. of BCD Characters into BCD Format	•	•	↕	↕	↕	③
Decrement	DEC							6A	7	2	7A	6	3				M − 1 → M	•	•	↕	↕	4	•
	DECA													4A	2	1	A − 1 → A	•	•	↕	↕	4	•
	DECB													5A	2	1	B − 1 → B	•	•	↕	↕	4	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A⊕ M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B ⊕M → B	•	•	↕	↕	R	•
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Acmltr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
Or, Inclusive	ORA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A+M → A	•	•	↕	↕	R	•
	ORB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B+M → B	•	•	↕	↕	R	•

Fig. 57 - 2° tableau des instructions Accumulateurs et mémoires

OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	IMPLIED OP	IMPLIED ~	IMPLIED #	Opérations logiques et arithmétiques	5 H	4 I	3 N	2 Z	1 V	0 C
Push Data	PSHA													36	4	1	A → M_{SP}, SP – 1 → SP	●	●	●	●	●	●
	PSHB													37	4	1	B → M_{SP}, SP – 1 → SP	●	●	●	●	●	●
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	●	●	●	●	●	●
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	●	●	●	●	●	●
Rotate Left	ROL							69	7	2	79	6	3				M	●	●	↕	↕	⑥	↕
	ROLA													49	2	1	A	●	●	↕	↕	⑥	↕
	ROLB													59	2	1	B (C ← b_7 ← b_0)	●	●	↕	↕	⑥	↕
Rotate Right	ROR							66	7	2	76	6	3				M	●	●	↕	↕	⑥	↕
	RORA													46	2	1	A	●	●	↕	↕	⑥	↕
	RORB													56	2	1	B (C b_7 → b_0)	●	●	↕	↕	⑥	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	●	●	↕	↕	⑥	↕
	ASLA													48	2	1	A (C ← b_7 b_0 ← 0)	●	●	↕	↕	⑥	↕
	ASLB													58	2	1	B	●	●	↕	↕	⑥	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	●	●	↕	↕	⑥	↕
	ASRA													47	2	1	A	●	●	↕	↕	⑥	↕
	ASRB													57	2	1	B (b_7 b_0 → C)	●	●	↕	↕	⑥	↕
Shift Right, Logic	LSR							64	7	2	74	6	3				M	●	●	R	↕	⑥	↕
	LSRA													44	2	1	A (0 → b_7 b_0 → C)	●	●	R	↕	⑥	↕
	LSRB													54	2	1	B	●	●	R	↕	⑥	↕
Store Acmltr.	STAA				97	4	2	A7	6	2	B7	5	3				A → M	●	●	↕	↕	R	●
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	●	●	↕	↕	R	●
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A – M → A	●	●	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B – M → B	●	●	↕	↕	↕	↕
Subtract Acmltrs.	SBA													10	2	1	A – B → A	●	●	↕	↕	↕	↕
Subtr. with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A – M – C → A	●	●	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B – M – C → B	●	●	↕	↕	↕	↕
Transfer Acmltrs	TAB													16	2	1	A → B	●	●	↕	↕	R	●
	TBA													17	2	1	B → A	●	●	↕	↕	R	●
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M – 00	●	●	↕	↕	R	R
	TSTA													4D	2	1	A – 00	●	●	↕	↕	R	R
	TSTB													5D	2	1	B – 00	●	●	↕	↕	R	R
																		H	I	N	Z	V	C

Fig. 58. — tableau de manipulation des instructions destinées au registre d'index et au stack pointer

		Modes d'adressage															Operations logiques et arithmetiques	Registre d'état					
		IMMED			DIRECT			INDEX			EXTND			IMPLIED				5	4	3	2	1	0
POINTER OPERATIONS	MNEMONIC	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$X_H - M, X_L - (M+1)$	•	•	(7)	↕	(8)	•
Decrement Index Reg	DEX													09	4	1	$X - 1 \rightarrow X$	•	•	•	↕	•	•
Decrement Stack Pntr	DES													34	4	1	$SP - 1 \rightarrow SP$	•	•	•	•	•	•
Increment Index Reg	INX													08	4	1	$X + 1 \rightarrow X$	•	•	•	↕	•	•
Increment Stack Pntr	INS													31	4	1	$SP + 1 \rightarrow SP$	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H, (M+1) \rightarrow X_L$	•	•	(9)	↕	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H, (M+1) \rightarrow SP_L$	•	•	(9)	↕	R	•
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M, X_L \rightarrow (M+1)$	•	•	(9)	↕	R	•
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M+1)$	•	•	(9)	↕	R	•
Indx Reg → Stack Pntr	TXS													35	4	1	$X - 1 \rightarrow SP$	•	•	•	•	•	•
Stack Pntr → Indx Reg	TSX													30	4	1	$SP + 1 \rightarrow X$	•	•	•	•	•	•

Légende :

OP → Code opération en hexadécimal (l'assembleur comprend le code hexadécimal et le langage mnémonique).

~ → Nombre de cycles du MPU

→ Nombre d'octets du programme d'instructions

+ → Signe arithmétique d'addition (plus)

— → Signe arithmétique moins

• → AND (ET) en algèbre de Boole

+ Inclusive OR

⊕ Exclusive OR — *En algèbre de Boole*

$\overline{M}$ Complément de M

→ Transférer dans

0 → Bit 0

00 → Octet 0

MSP → Emplacement mémoire pointé — par le stack pointer. Les instructions du mode d'adressage accumulateur sont incluses dans la colonne d'adressage impliqué (implied).

Le Registre d'Etat ***(CCR).*** Condition Code Register : Symboles utilisés dans ce registre.

H (Half carry) → voir paragraphe 5 du présent chapitre

I (Interrupt Mask Bit)

N (Bit de signe)

Z (passage à zéro accumulateur)

V (OVERFLOW - Bit de dépassement et complément à 2)

Idem voir paragraphe 5 du présent chapitre : registre d'état : les indicateurs.

C (carry Flip-Flop)

R → Reset (Remise en route)

S → Set (Placé)

Voir paragraphe 7 c

↕ Si cela est exact le tester et le placer, sinon l'effacer.

• l'instruction peut le tester, mais ne peut pas modifier le bit du flip-flop (point mémoire personnalisé par sa lettre du registre d'Etat).

Fig. 59. — Tableau des instructions de branchements et de sauts. (« branch » et « Jump »).

		Modes d'adressage												Tests de branchements	Registre d'état					
		RELATIVE			INDEX			EXTND			IMPLIED				5	4	3	2	1	0
OPERATIONS	MNEMONIC	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Branch If ⩾ Zero	BGE	2C	4	2										N ⊕ V = 0	•	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										Z + (N ⊕ V) = 0	•	•	•	•	•	•
Branch If Higher	BHI	22	4	2										C + Z = 0	•	•	•	•	•	•
Branch If ⩽ Zero	BLE	2F	4	2										Z + (N ⊕ V) = 1	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										N ⊕ V = 1	•	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2											•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3					•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										02	2	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1		(10)					
Return From Subroutine	RTS										39	5	1		•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1		•	•	•	•	•	•
Wait for Interrupt	WAI										3E	9	1		•	(11)	•	•	•	•

Fig. 60 — Tableau de manipulations des instructions destinées au registre d'état.

OPERATIONS	MNEMONIC	Mode Adressage IMPLIED OP	~	#	Opération booléenne	Registre d'état 5 H	4 I	3 N	2 Z	1 V	0 C
Clear Carry	CLC	0C	2	1	0 → C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	0 → I	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	0 → V	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	1 → C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	1 → I	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	1 → V	•	•	•	•	S	•
Acmltr A → CCR	TAP	06	2	1	A → CCR	(12)					
CCR → Acmltr A	TPA	07	2	1	CCR → A	•	•	•	•	•	•

Note. — Sur les registres d'état dans les différents tableaux de manipulations d'instructions :

Chiffres entourés de (1) *à* (9) *→ Le bit de l'indicateur spécifié dans la colonne correspondante du registre d'Etat est testé (voir MC6800 - Programming Manual)*

(10) *→ Charge venant du stack pointer dans le registre d'état.*

(opérations spéciales. Voir MC6800. Programming manual).

(11) *→Le bit est placé lorsque survient une demande d'interruption s'il était placé antérieurement, le signal NMI serait indispensable pour rompre l'état d'attente* ***(WAI)****.*

(12) *→ A indiquer suivant le contenu de l'accumulateur A.*

(13) *→ Les bits 6 et 7 dans le registre d'état sont en permanence au niveau logique « 1 » (voir figure n° 61).*

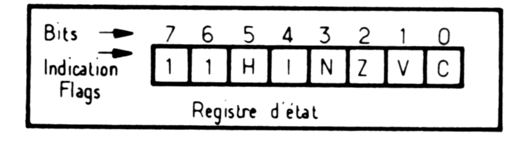


Figure 61.

CHAPITRE **9**

Évolution technique des microprocesseurs et des micro-ordinateurs.

1. Introduction

Cette évolution des microprocesseurs a porté sur les 3 axes de besoins différents suivants :

a) Vers une intégration totale des circuits existants.

En vue d'une nouvelle génération miniaturisée de micro-ordinateurs. Ainsi le microprocesseur **MC6802** associé au composant d'interface **MC6846** vont tous deux constituer à eux seuls un véritable micro-ordinateur aussi puissant que celui bâti précédemment autour du **MC6800.**

Mieux encore, sur une seule « puce » (en anglais : one chip) le **MC6801** va admettre toutes les possibilités d'un véritable micro-ordinateur.

b) Vers la réalisation très économique de problèmes simples.

Un micro-ordinateur peut maintenant être réalisé à l'aide d'un ***microprocesseur monobit*** très simple et très bon marché à ***16 broches de sorties*** seulement.

Il peut aisément résoudre des problèmes de prises de décisions applicables aux domaines de l'électro-ménager (Exemple : programmation d'une machine à laver) et de l'industriel (programmation d'automatismes en vue de la commande)

a) des séquences dans une machine transfert
b) de commutations et multiplexages divers
c) d'intervalles de temps..., etc.

Nous en décrivons un exemple précis (le microprocesseur **MC14500B**) dans notre livre : **Le microprocesseur en action.**

c) Vers le développement des circuits très sophistiqués.

Ces circuits présentent de multiples possibilités de résolution ***« en nombre et en finesse de précisions »*** grâce au triomphe de la nouvelle technologie **VLSI** (very large scale integration). Nous allons décrire dans ce chapitre, comme exemple, le **MC68000** de MOTOROLA.

Il s'agit ***d'un microprocesseur monolithique de technologie HMOS encapsulé dans un boîtier DIL à 64 broches.***

A la fin de ce chapitre nous dirons, pour le terminer, quelques mots sur la conception originale des microprocesseurs bipolaires montés en ***« tranches de 4 bits ».***

2. Vers l'intégration totale des circuits

A) Le microprocesseur **MC6802** est essentiellement constitué d'un MC6800. Il conserve toutes les qualités de ce dernier, en particulier, il a le même répertoire d'instructions et de modes d'adressages. Sa compatibilité est totale avec tous les circuits périphériques du MC6800.

Cependant il intègre en plus des circuits du MC6800, sur la même pastille :

Une horloge
et 128 Octets de RAM (voir figure 62).

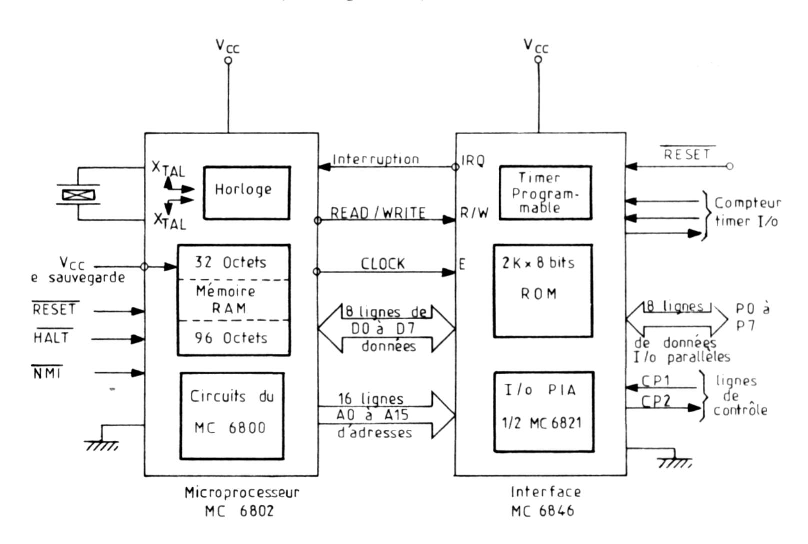


Fig. 62. *Un micro-ordinateur à 2 circuits.*

L'horloge incorporée peut fonctionner à une fréquence maximale de 1 MHz, cependant le circuit comprend un diviseur par 4, ce qui permet d'utiliser un quartz externe de 4 MHz plus économique qu'un quartz 1 MHz.

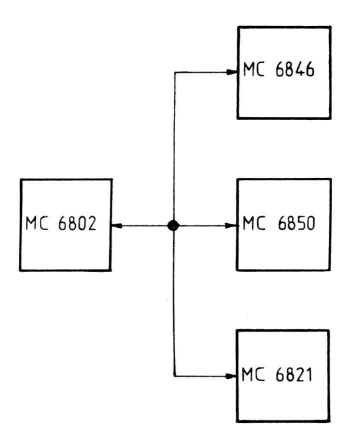


Fig. 63. *A l'aide d'un « Kit » de 4 circuits on peut composer ce synoptique représentant un ensemble de caractéristiques très voisines de celles du MC6801.*

Parmi les 128 octets de RAM incorporés, ***les 32 premiers octets de cette mémoire disposent d'un mode à faible consommation et peuvent ainsi être branchés sur une alimentation de secours*** permettant en cas de coupure de l'alimentation principale de sauvegarder les informations essentielles au bon fonctionnement du microordinateur.

Lorsque le **MC6802** ***est couplé avec le circuit d'interface*** **MC6846** ***il réalise une configuration minimale de micro-ordinateur*** (voir figure 62).

Rappelons qu'une configuration identique avec un MC6800 demande au moins l'usage de 5 circuits intégrés.

Le MC6846 est un composant d'interface de la famille MC6800. Il comprend trois parties différentes :

a) ***Une ROM de 2048 octets,*** ce qui est suffisant à une programmation minimale. Cette mémoire est programmable par masque selon les spécifications indiquées par le client.

b) ***Un compteur temporisateur à 16 bits*** qui est également programmable. Il peut ainsi compter des événements, mesurer des fréquences et des intervalles de temps, générer des ondes rectangulaires..., etc.

c) ***Un interface bidirectionnel de données 8 bits*** (équivalent à 1/2 - PIA - MC6821) avec 2 lignes de contrôle CP1 et CP2.

B) Le microprocesseur **MC6801** est un circuit monolithique qui constitue en lui-même un véritable micro-ordinateur.

En effet, son architecture (voir synoptique de la figure 63) comporte :

1. ***Tous les circuits de base du microprocesseur MC6800.***

2. ***L'oscillateur d'horloge incorporé ainsi que la RAM de 128 octets du MC6802.***

3. ***La ROM de 2Koctets et le temporisateur*** (en anglais : timer) ***16 bits du MC6846.***

4. ***Une large gamme de possibilités d'entrées/sorties parallèles et programmables*** équivalente au circuit PIA - MC6821, que nous étudierons au chapitre 11. Cette gamme, jusqu'à 31 lignes d'I/0, permet de gérer les périphériques externes.

5. ***Un accès série,*** pour les équipements de communications, équivalent à l'ACIA - MC6850 que nous étudierons au cours du chapitre 11.

6. ***Un système d'interruptions à 8 niveaux*** augmentant ainsi par rapport au MC6800, les performances d'emploi du micro-ordinateur.

7. Enfin, dans le cas d'applications nécessitant une capacité supplémentaire, il conserve une totale compatibilité avec les circuits de la famille du MC6800 et peut ainsi disposer ***d'une possibilité d'extension mémoire jusqu'à 65Koctets.***

3. Vers la réalisation très économique de problèmes simples

La figure 64 nous donne un aperçu sur le système minimum de fonctionnement d'un micro-ordinateur bâti autour de l'unité de contrôle industriel **(UCI). MC14500B.** Il s'agit d'un circuit **CMOS.** Les données

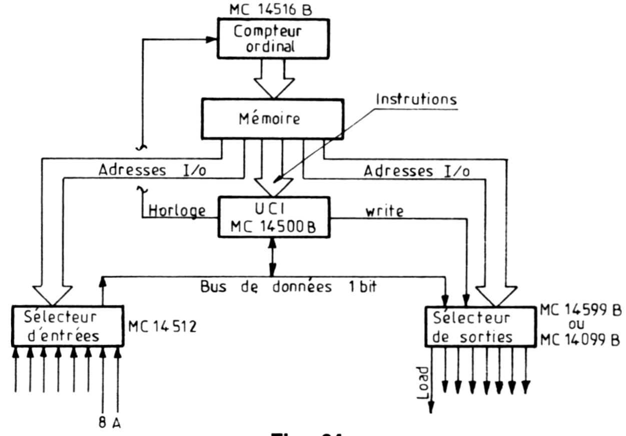


Fig. 64

d'entrées sont traitées et sorties au niveau de mots d'un seul bit et c'est pourquoi nous pouvons affirmer qu'à ce niveau nous sommes en présence ***d'un microprocesseur monobit.*** Par contre, ***les instructions, au nombre de 16 au total, sont bâties à l'aide de mots de 4 bits.***

Tous les circuits intégrés mis à part la mémoire qui composent le système minimum sont des circuits de technologie CMOS encapsulés dans des boîtiers bon marché simples DIL à 16 broches de sorties.

Nous avons énoncé rapidement dans l'introduction à ce chapitre les utilisations possibles de ce système.

Il nous reste à en tracer le principe de fonctionnement.

Après avoir ***chargé le programme en mémoire,*** une séquence de celui-ci se déroule de la manière suivante :

Le compteur ordinal **MC14516B** extérieur au microprocesseur ***pointe les adresses*** inscrites en mémoire à chaque nouveau cycle d'horloge délivré et commandé par le MC14500B.

La mémoire va alors rechercher l'adresse de la donnée d'entrée du **MC14512** ***à sélectionner.*** Celle-ci est ensuite introduite ainsi que l'instruction désignée dans **l'ALU de l'UCI.**

L'opération de logique désirée est effectuée dans l'ALU avec une deuxième donnée d'entrée préalablement sélectionnée.

La mémoire fournit alors à l'UCI ***l'adresse de la sortie de charge LOAD adéquate*** et le résultat réalisé dans l'ALU est effectivement adressé à cette sortie.

Le livre « Le microprocesseur en action » décrit en détails la mise en mémoire et le déroulement des programmes d'un microsystème bâti autour de l'UCI - MC14500B.

4. Vers le développement des circuits très sophistiqués

a) Le MC 68000

1. Technologie du microprocesseur

Ce microprocesseur a pû être réalisé chez MOTOROLA grâce à la nouvelle famille technologique ***V.L.S.I.*** (very large scale integration, c'est-à-dire : Très large échelle d'intégration). Par rapport à la famille d'intégration ***L.S.I.*** on peut maintenant diffuser, sur une seule pastille de silicium de quelques dizaines de millimètres carrés ***dix fois plus de jonctions actives,*** d'où l'idée de comparaison dans son appellation entre l'ancien microprocesseur MC6800 et le nouveau MC68000. La technolo-

gie propre à ce nouveau microprocesseur à pris le nom de ***H.M.O.S.*** (high MOS density c'est-à-dire MOS à haute densité). Entre le H.M.O.S. et le N.M.O.S. on peut établir les comparaisons suivantes en ce qui concerne les caractéristiques :

a) La densité de circuit, sur la pastille de silicium, est plus de 2 fois plus importante. Dans le MC68000 une cellule élémentaire de logement d'un « bit » occupe une surface de ***1852 microns-carré*** alors que dans le MC6800 elle était de ***4128 microns-carré.***

b) Le produit : ***Durée de propagation X puissance consommée*** est 4 fois meilleur en technologie H.M.O.S.,
soit en : ***N.M.O.S. = 4 picojoules***
et en : ***H.M.O.S. = 1 picojoule***

2. Architecture interne et fonctionnement du microprocesseur.

Le microprocesseur MC68000 peut exploiter 6 sortes de données :

a) ***le binaire naturel***
b) ***le décimal codé en mots de 4 bits*** (système BCD — Voir tableau page 33)
c) ***les caractères ASCII*** (voir page 140)
d) ***les mots de 8 bits*** (bytes)
e) ***les mots de 16 bits*** (un mot de 16 bits est appelé — ***WORD***)
f) ***Les mots de 32 bits*** (travaillant en double longueur — Un mot de 32 bits sera appelé ici : ***Long word***).

Le microprocesseur MC68000 comporte un jeu de ***61 instructions*** distinctes écrites en langage mnémonique de 3 ou 4 ou même 5 lettres. Parmi ces instructions, on peut, en résumé, citer celles :

a) ***De fonctions arithmétiques signées ou non signées*** comprenant notamment les 4 opérations fondamentales (Addition, soustraction, multiplication, division) y compris les instructions auxiliaires servant à les déterminer telles que : complément à « 1 », complément à « 2 », décalage dans un registre des bits à gauche ou à droite, etc.
b) ***De fonctions logiques*** telles que : ET, OU inclusif, OU exclusif.
c) ***De branchements, sauts à sub-routines, d'initialisation, d'arrêt, de charge des différents registres du microprocesseur, de test des indicateurs de son registre d'état, etc.***
d) De nouvelles instructions, en particulier ***TRAP*** et ***TRAPV.*** Elles doivent être utilisées par le programmeur dans un but d'applications destinées à la détection d'erreurs ou bien encore à celui de correction de routines.

Ces 61 instructions sont réparties en 5 modes principaux d'adressage (Voir à partir de la page 87).

Ce sont les modes :

adressage direct
adressage indirect
adressage immédiat
adressage absolu
adressage relatif

Pour plus de souplesse d'emploi dans le MC68000, chacun de ces 5 modes principaux d'adressage comporte des particularités distinctes, ce qui fait que l'on pourrait en réalité parler de ***14 sous-modes d'adressage de ce microprocesseur.***

A noter enfin qu'un « ***Trace mode*** » signalé par un bit de test dans le registre d'état donne la possibilité au programmeur de vérifier son programme au cours même de son élaboration sucessivement instruction par instruction.

Nous en arrivons à l'architecture interne proprement dite du microprocesseur MC68000 (Voir figures 65 et 66).

Le MC68000, comme le montre la figure 65, comprend : ***16 registres de 32 bits*** en plus du ***compteur ordinal de 24 bits*** et du ***registre d'état de 16 bits.***

— ***(D0 à D7)*** sont les 8 premiers registres destinés au stockage et à la lecture des données soit sous forme ***de mots de 8 bits*** (bytes)
— ou bien encore ***de mots de 16 bits*** (word)
— ou même enfin ***de mots de 32 bits*** (longword)

— ***(A0 à A7)*** sont les 8 seconds registres destinés à l'adressage des données ou bien encore au registre de sauvegarde « ***Stack pointer*** » (Voir page 81). Ils peuvent être utilisés soit avec ***des mots de 16 bits*** ou bien encore ***des mots de 32 bits.***

D'autre part les 24 bits du compteur ordinal fournissent un adressage mémoire d'une capacité de ***16 méga-bytes*** (16.777.216 mots de 8 bits).

Examinons enfin (figure 65) le registre d'état. Il comporte :

a) 8 niveaux possibles d'interruptions (c'est-à-dire 2^3 bits) grâce au jeu de ses ***3 indicateurs*** (flags) I_0 - I_1 - I_2.

b) Les codes de conditions déjà connus :

Overflow (V)
Zéro (Z)
Négative (N)
Carry (C)
Extend (X)

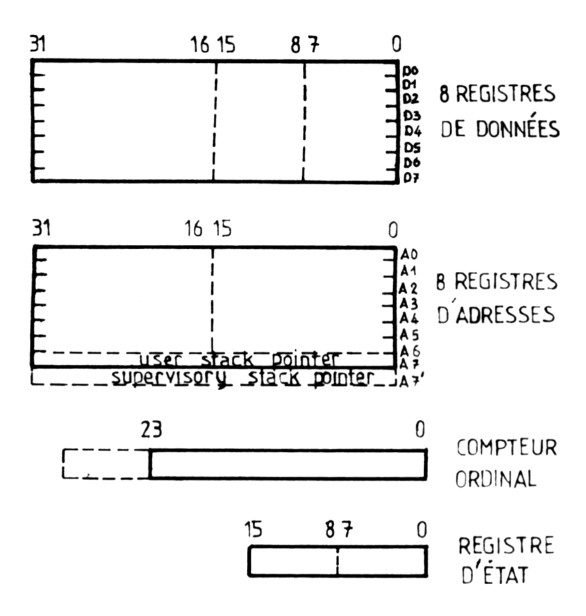


Figure 65 - registres du MC 68000

L'indicateur ***X*** est un bit de test complémentaire du bit carry ***C*** plus spécialement chargé dans une instruction « Link » de rotation de relier le MSB et le LSB (Voir fig. 44 — page 74).

c) Le nouveau bit de Test ***Trace (T)*** dont nous venons de voir la signification.

d) L'état de supervision : ***Supervisory (S)***

Le « ***TRACE MODE*** » ne peut être demandé par le programmeur que lorsque le microprocesseur est dans l'état de supervision : ***(Supervisory - S)*** et non dans l'état habituel de fonctionnement appelé : « ***USER*** ».

e) Enfin la figure 66 montre ***des places disponibles d'indicateurs (5,6,7*** puis ***11, 12*** et enfin ***14***) dans le registre d'état en vue d'améliorer les possibilités d'exploitation du MC68000 lors d'extensions futures prévues quant à la famille M68000.

Le MC68000 est compatible avec les périphériques du microprocesseur MC6800, en particulier avec ceux que nous avons déjà examinés

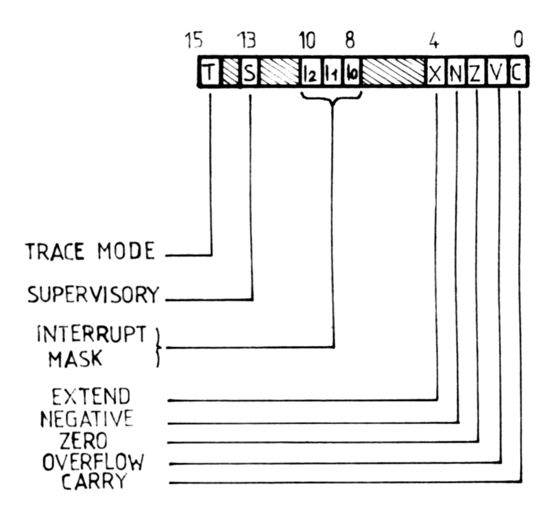


Figure 66 - registre d'état

c'est-à-dire : ***Le MC6821*** (Voir à partir de la page 131) et le ***le MC6850 (ACIA*** Voir à partir de la page 134).

Ajoutons également les ***MC 6843*** et ***MC6849*** qui sont des nouveaux circuits de contrôle des ***Floppy Disc*** (voir pages 139 et 157) et le ***MC 6847*** qui est un circuit générateur de caractères d'affichage vidéo sur un écran de TV et dont nous reparlerons plus en détail au cours du sous-chapitre consacré aux consoles de visualisation (moniteurs vidéo-fréquences).

3. Configuration des broches de sorties du microprocesseur MC68000 (Voir figure n° 67).

La lecture du schéma de la figure 67 nous permet de distinguer :

— Un « ***Bus d'adresses*** » unilatéral de 23 lignes parallèles A_1 ***à*** A_{23} et un « ***Bus de données*** » de 16 lignes parallèles bilatérales D_0 ***à*** D_{15}. Le bus de données étant séparé du bus d'adressage cela n'oblige pas à définir un circuit de multiplexage entre les adresses et les données.

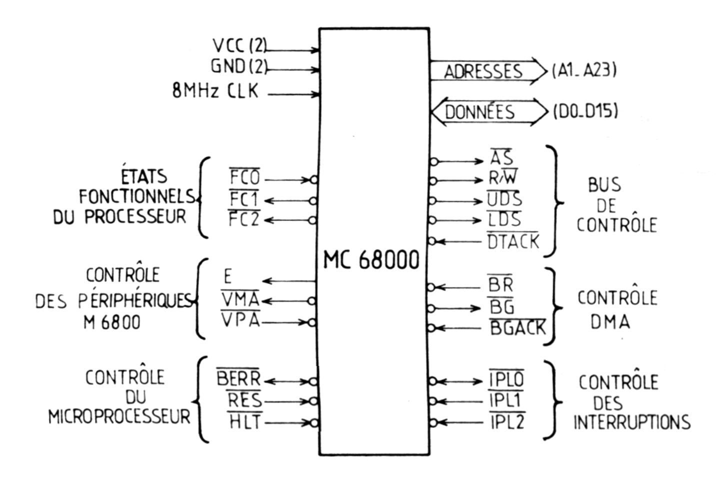


Figure 67

— 2 connections sont réservées à l'application, de la tension + de l'alimentation ***(+ Vcc = 5 Volts)*** et 2 autres à la tension de référence ***(GND = retour à la masse).*** La valeur de la tension d'alimentation nous montre la ***compatibilité,*** avec les composants périphériques de ***la famille TTL.***

— Un « Signal d'Horloge » ***(CLK = clock)*** de ***8 MHz.*** Il peut être obtenu à partir d'un oscillateur à quartz.

— Un bus de contrôle de 5 lignes — 4 lignes sont orientées du microprocesseur vers des mémoires ou des I/O d'interfaces. Ce sont :

a) la ligne ***$\overline{\text{AS}}$ (Adress Strobe)*** qui permet d'affirmer que l'adresse sur le bus d'adresses est validée.

b) la ligne ***R/$\overline{\text{W}}$ (read-write)*** qui permet de distinguer un cycle de lecture ou un cycle d'écriture d'une mémoire.

c) les 2 lignes ***$\overline{\text{UDS}}$ (Upper data strobe)*** et ***$\overline{\text{LDS}}$ (Low Data Strobe)*** qui permettent par exemple dans un mot de 16 bits de distinguer l'octet de plus faible poids ***$\overline{\text{LDS}}$*** de l'octet de plus fort poids ***$\overline{\text{UDS}}$.***

La 5e ligne ***$\overline{\text{DTACK}}$ (data transfer knowledge)*** intime par exemple l'ordre de transférer la donnée lors d'un cycle de lecture de la mémoire vers le microprocesseur.

— 3 lignes de contrôle des interruptions ***$\overline{\text{IPL0}}$ - $\overline{\text{IPL1}}$ - $\overline{\text{IPL2}}$*** qui, dans un système binaire, permettent à l'aide des bits 1 et 0 la possibilité de 2^3 soit 8

niveaux différents de priorité de demandes d'interruptions d'un programme en cours.

— 3 lignes qui dans un même ordre d'idées permettent à l'aide de 8 combinaisons binaires possibles de définir l'état fonctionnel, du microprocesseur. Dans le but d'une meilleure compréhension dans le tableau de la figure 68 ci-dessous nous donnons en fonction de la composition binaire de ***$\overline{FC2}$ - $\overline{FC1}$ et $\overline{FC0}$*** les 8 déterminations opérationnelles du microprocesseur.

Tableau figure 68

$\overline{FC2}$	$\overline{FC1}$	$\overline{FC0}$	Code de fonction	Repère
0	0	0	Other	①
0	0	1	User program	②
0	1	0	User data	③
0	1	1	Halt	④
1	0	0	Test mode	⑤
1	0	1	Supervisor program	⑥
1	1	0	Supervisor data	⑦
1	1	1	Interrupt knowledge	⑧

Commentaires sur ce tableau — Le microprocesseur est dans l'état arrêté ***(HALT)*** — lorsque est logé :

a) le bit ***0*** dans le registre ***$\overline{FC2}$***
b) le bit ***1*** dans le registre ***$\overline{FC1}$***
c) le bit ***1*** dans le registre ***$\overline{FC0}$.*** (Voir④).

Pour les autres états décelables de la même manière se reporter au tableau de la figure 68. Ce sont :

L'état de supervision ou l'état usuel de fonctionnement soit au cours du déroulement d'un programme ou soit lors du transfert d'une donnée — ②③⑥⑦.

Le programmeur peut également à un moment précis rechercher le mode de fonctionnement du microprocesseur ***(Test mode)*** ⑤.

On peut encore désirer connaître un niveau d'interruption 

Enfin lorsque les 3 registres sont à zéro le microprocesseur est dans un autre état ***(OTHER)*** ① que ceux qui viennent d'être précédemment décrits.

— 3 lignes de ***contrôle en DMA*** assurent la demande d'un périphérique de ***fonctionnement en accès direct à la mémoire*** sans être obligé de passer par le microprocesseur. Ces 3 lignes sont utilisées dans l'ordre suivant :

$\overline{BR}$ ***(bus request)*** est la demande du périphérique au microprocesseur d'accès direct à la mémoire.

$\overline{BG}$ ***(bus grant)*** est la réponse affirmative que peut donner le microprocesseur au périphérique.

$\overline{BGACK}$ est l'allocation du Bus permettant l'échange de données entre le périphérique et la mémoire.

— 3 lignes de contrôle du microprocesseur définissent :

a) La commande d'arrêt du microprocesseur ***(***$\overline{HLT}$ = ***Halt).***

b) La période d'initialisation ou remise à zéro du microprocesseur ***(***$\overline{RES}$ = ***Restart).***

c) La mise en évidence d'une erreur sur un bus ***(***$\overline{BERR}$ = ***bus error).***

Enfin 3 lignes de contrôle du microprocesseur avec des périphériques de la famille M6800. Ce sont :

a) l'indication ***E = enable*** par le microprocesseur d'un bus disponible

b) la validation $\overline{VMA}$ = ***Valid memory adress*** au moment du transfert d'une adresse du microprocesseur vers la mémoire de la famille M6800.

c) La validation $\overline{VPA}$ = ***Valid peripheral adress*** au moment du transfert d'une adresse du périphérique (composant de la famille M6800) au microprocesseur.

4. Fonctionnement d'un cycle de lecture.

(Tableau figure 69)

5. Fonctionnement d'un cycle écriture (tableau figure 70)

Ces 2 tableaux figures 69 et 70 nous donnent un exemple assez simplifié mais cependant significatif du fonctionnement logiciel entre le microprocesseur MC68000 et ses mémoires ou circuits d'interface associés au cours d'un cycle de lecture et d'un cycle d'écriture.

Tableau figure 69 (cycle lecture)

Microprocesseur	Mémoire ou interface
1) Adresse de la mémoire 1-1 Placer l'adresse sur A_1 à A_{23} 1-2- Placer R/$\overline{W}$ sur Read - 1-3- Valider le bus d'Adresse $\overline{AS}$ 1-4- Valider les données $\overline{UDS}$ et $\overline{LDS}$	*2) Entrée des données* 2-1- Décoder l'adresse 2-2- Placer les données sur D0 à D15 2-3- Intimer l'ordre de transfert des données ($\overline{DTACK}$).
3) Acquisition des données 3-1- Charger les données dans le registre interne adéquat. 3-2- Effacer signal Validation $\overline{UDS}$ et $\overline{LDS}$ 3-3 Effacer signal Validation AS	*4) Fin du cycle de lecture* 4-1- Supprimer données sur D0 à D15 4-2- Supprimer ordre de transfert des données (DTACK)
5) Prêt pour départ prochain cycle	

Tableau figure 70 (cycle écriture)

Microprocesseur	Mémoire ou Interface
1) Acquisition de données 1-1- Placer R/$\overline{W}$ à write 1-2- Placer les données sorties de l'Alu sur D0 à D15 1-3- Valider $\overline{UDS}$ et $\overline{LDS}$	*2) Acceptation des données* 2-1- Charger les données D0 à D15 comme définis par $\overline{UDS}$ et $\overline{LDS}$ 2-2 Valider le transfert de données $\overline{DTACK}$.
3) Terminer le transfert en sortie 3-1- Effacer la validation UDS et LDS 3-2- Supprimer les données sur D0 à D15 3-3 Placer R/$\overline{W}$ sur Read	*4) Terminaison du cycle d'écriture* 4-1- Supprimer la validation $\overline{DTACK}$.
5) Prêt pour départ prochain cycle	

b) Le MC68008

Il s'agit d'un microprocesseur d'architecture interne identique au MC68000 mais d'organisation externe comportant seulement un **bus de données de 8 bits** (D0 à D7). Il est donc prévu pour fonctionner normalement en simple précision sur des mots de 8 bits tout en profitant des possibilités techniques de finesse d'élaboration des programmes du MC68000.

c) Le MC68020

Il s'agit cette fois-ci d'un véritable **microprocesseur de mots de 32 bits** actuellement en développement.

5. Les microprocesseurs montés en tranches de 4 bits

Il s'agit ici de **microprocesseurs bipolaires** tel que le **MC10800** (technologie ECL) ou le **AM2901** (technologie TTL).

Rappelons que les circuits MOS sont plus simples à fabriquer et moins coûteux que les circuits de technologie bipolaire, par contre, ***la technologie bipolaire offre un avantage incontestable dans de nombreuses applications sur le plan de la vitesse d'exécution*** notamment s'il est question d'effectuer des réponses en temps réel sur des calculs complexes.

Une séquence d'instructions s'effectue à une moyenne de 2 microsecondes avec un microprocesseur N.MOS, avec un microprocesseur bipolaire ECL, cette séquence peut ne demander qu'un temps de réalisation de 0,1 microseconde.

Dans le domaine des micro-ordinateurs l'approche bipolaire est bien différente de celle des circuits MOS.

On réalise ici un concept de micro-ordinateur et de microprocesseurs débité en « ***tranches de bits*** ».

Tous les composants LSI, y compris le microprocesseur, peuvent être disposés en cascade afin de réaliser des systèmes qui seront plus ou moins complexes.

Ainsi avec des tranches élémentaires de 4 bits de circuits disposés en cascade on pourra fabriquer des microprocesseurs et des micro-ordinateurs 8 bits, 16 bits et 32 bits.

Si les microprocesseurs MOS sont programmés à l'aide d'un logiciel qui leur est propre, ***les familles bipolaires peuvent généralement utiliser le logiciel de n'importe quel ordinateur existant.***

CHAPITRE **10**

Les mémoires

Elles servent à stocker les données nécessaires à la programmation d'instructions et d'opérandes qui seront traitées dans l'unité centrale du microprocesseur c'est-à-dire dans son ALU.

Elles sont utilisées à 2 niveaux :

a) Au niveau des périphériques :

Ici, elles sont nécessaires à l'Etablissement d'un programme à partir d'un périphérique qui peut être disposé loin du micro-ordinateur interrogé.

Ce sont généralement des mémoires de *forte capacité.*

Dans cette catégorie, on distingue :
— les mémoires à ***cartes perforées***
— les mémoires à ***bandes perforées***
— Les mémoires à ***disques (floppy-disk)***
— les mémoires à ***bandes magnétiques.***

Ce sont des mémoires relativement lentes car pour accéder à une information précisée, il faut faire défiler la bande précédant l'inscription désirée.

On dit, dans ce cas que l'on a affaire à une mémoire à ***accès séquentiel.***

Toutes ces mémoires portent le nom général de ***mémoires de masse.***

b) Au niveau de l'unité centrale :

A l'origine, il s'agissait de ***mémoires à tores.*** A cause de leur fabrication mécanique assez compliquée et de leur faible niveau de tension utilisable (quelques millivolts), elles ont été progressivement remplacées aujourd'hui par des mémoires à circuits intégrés.

Dans la plupart de ces derniers circuits, chaque donnée figure à une adresse connue que l'on peut atteindre directement. Ce sont donc des mémoires d'accès plus rapides que les précédentes. On dit encore que ce sont des mémoires à ***accès aléatoire.***

Il en existe de nombreuses catégories dont nous allons donner par la suite la signification.

On les appelle des ***mémoires centrales.***

Nous en avons déjà parlé au cours des chapitres précédents, notamment, d'une manière générale dans le chapitre 2 et d'une façon plus détaillée dans le chapitre 7.

Notre exposé étant axé sur la micro-informatique, nous allons parler plus longuement, maintenant, des mémoires centrales à circuits intégrés ***(LSI).*** Nous aborderons en :

1. Les caractéristiques et les définitions des paramètres électriques des mémoires à circuits intégrés.

2. Les diverses catégories et les fabrications de mémoires à circuits intégrés.

3. Les diverses mémoires, à titre d'exemple, de la famille MOTOROLA M6800.

1. Caractéristiques et définitions des paramètres électriques des mémoires à circuits intégrés.

Les mémoires à circuits intégrés sont : ***volatiles.***

On dit qu'une mémoire est volatile quand elle perd son information lorsque survient une coupure de sa tension d'alimentation. On peut dire par contre à l'inverse qu'une mémoire inscrite sur bande magnétique est ***non volatile.***

Une mémoire est ***destructive*** lorsque l'opération de lecture détruit l'information.

Certaines mémoires à circuit intégré sont destructives volontairement. C'est le cas des ***mémoires vives (RAM*** = Random Accès Memory).

D'autres sont ***indestructibles*** c'est le cas des ***mémoires mortes (ROM*** = Read Only Memory).

Lorsque le CPU a besoin d'une donnée située dans une mémoire, il va la chercher par l'intermédiaire de son « BUS d'Adresses » à l'adresse à laquelle, elle est logée dans la mémoire. Une fois l'adresse jointe, il lit la donnée. De deux choses, l'une :

Ou bien il s'agit d'une mémoire morte et après lecture, la donnée est ***conservée*** a son adresse, ou bien il s'agit d'une mémoire vive et la donnée est ***détruite*** après sa lecture. Cela laisse la possibilité d'un emplacement pour une donnée nouvelle qui pourrait être calculée par l'ALU du microprocesseur et stockée à la place vide en mémoire. Le ***transfert*** - micro-

processeur → mémoire se réalise à l'aide du bus des données (DATA BUS).

Les caractéristiques de temps d'inscription et de lecture dans les mémoires sont assez complexes, aussi, afin de mieux les définir, nous allons nous reporter aux figures 71 et 72.

Figure 71. Cycle d'écriture-mémoire

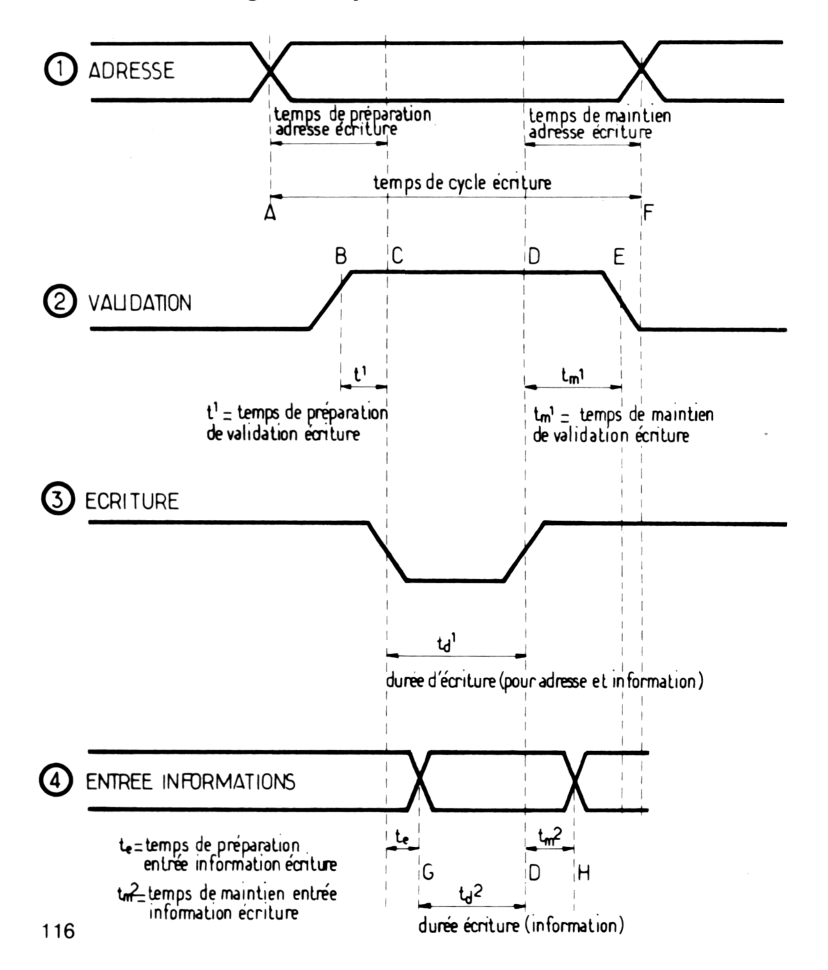


Figure 71. ***Le cycle d'écriture-mémoire.***

Ce cycle ① dans une mémoire synchrone est défini par le cadencement du signal d'horloge ***Q.*** Sa durée est donc fonction de la fréquence du quartz de l'oscillateur. (Exemple : un quartz de 1 MHz va déterminer un temps de cycle de 1 microseconde).

En ②, c'est le signal ***VMA*** (validation du bus d'adresses à la mémoire).

En ③ c'est le ***temps réel d'écriture des bits*** d'adresse en premier lieu, puis d'informations en second lieu dans la mémoire.

En ④, c'est ***l'introduction de l'information*** à l'aide du bus de données. Cette introduction va donc se réaliser à *la suite* de l'écriture de l'adresse. L'écriture de l'adresse s'effectue pendant le ***temps de préparation d'entrée de l'information (te).***

D'autre part, la figure montre encore que chaque écriture d'adresse et d'information se conçoit en 3 temps.

Ainsi pour l'écriture d'une information (donnée) on dispose de :

a) Un ***temps de préparation « te »*** à l'écriture
b) Un ***temps d'écriture « td_2 »*** proprement dit
c) Un ***temps de maintien « tm_2 »*** d'écriture.

Enfin le temps de ***validation*** d'écriture dans la mémoire s'inscrit à l'intérieur du temps de ***cycle d'écriture*** c'est-à-dire à l'intérieur des temps de préparation pour une part et de maintien d'écriture pour une autre part.

Figure 72. ***Le cycle de lecture mémoire***

Nous retrouvons les mêmes configurations de signaux que dans la figure précédente concernant ① ② et ③

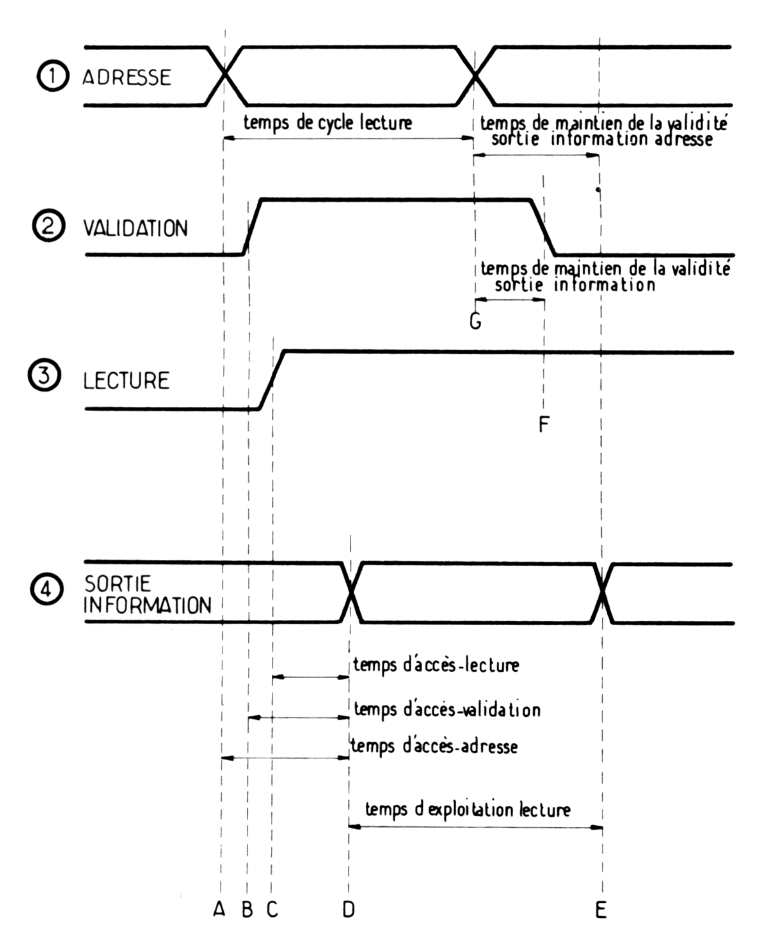
③ Peut être défini par le signal ***read/write*** en position ***read,*** venant du microprocesseur, en figure 71, ce signal devrait être commuté sur la position : ***write.***

④ En ce qui concerne ce synoptique, nous avons en plus, par rapport à celui de la figure 71, les définitions des différents temps d'accès.

Temps d'accès-lecture. C'est le temps compris entre l'instant de disponibilité du signal de lecture (C) et celui du début de lecture de l'information à l'adresse désirée (D) [temps CD].

Temps d'accès-validation. C'est le temps compris entre l'instant de disponibilité du signal de validation (B) et celui du début de lecture de l'information à l'adresse désirée (D) [temps BD].

Figure 72. Cycle de lecture-mémoire



Les autres caractéristiques intéressantes d'une mémoire sont :

Sa capacité : elle s'exprime par le nombre total de mots de « n » bits que la mémoire peut stocker.

Sa cadence de transfert : Elle mesure en fréquence c'est-à-dire en « n » bits par seconde ce que la mémoire peut au total accepter d'informations (soit en lecture ou bien encore soit en écriture).

2. Les diverses catégories et leur fabrication de mémoires à circuits intégrés.

Ces catégories de mémoires à circuits intégrés sont nombreuses. Nous ne citerons que celles qui nous semblent les plus importantes parmi tous les types technologiques qui fleurissent à chaque période nouvelle de temps.

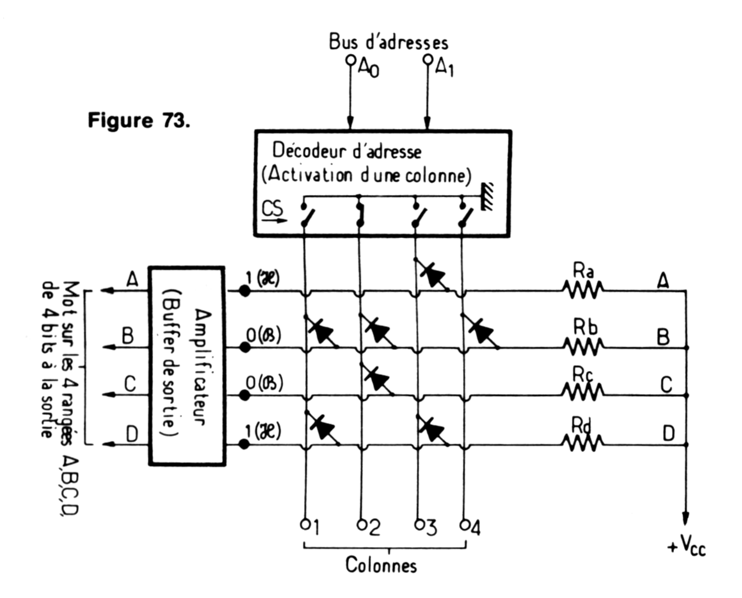
Les ***RAM*** (Random Accès Memory). Mémoires vives à accès aléatoire. On peut les lire, les écrire, les effacer. Il existe des ***RAM statiques*** et des ***RAM dynamiques.*** Nous vous invitons à vous reporter au chapitre 7 où nous les avons déjà décrites.

Les ***ROM*** (Read Only Memory). Mémoires mortes non destructives. Ce sont des mémoires réservées exclusivement à la lecture. Elles sont à accès aléatoire. On peut distinguer des ***ROM à diodes*** et des ***ROM à transistors*** bipolaires ou MOS. Ces mémoires sont dites : ***mémoires figées,*** enregistrées une fois pour toutes, et, à la demande du client, par le fabricant. Ainsi, dans l'exemple de ROM à diodes que nous présentons à la figure 73, le ***masque de diffusion*** a masqué 9 points programmés par le client parmi les 16 points possibles de mémoire. Lors de l'opération de diffusion, seul (16-9) = 7 jonctions diodes seront établies. *Les jonctions diodes diffusées* permettent de refermer la tension d'alimentation + VCC à la masse et d'inscrire sur la rangée correspondante le bit « O » (niveau $\mathscr{B}$) à condition qu'une des colonnes soit activée par un signal provenant d'une des entrées d'activation, du ***décodeur d'adresse,*** appelées respectivement : CS0, CS1, CS2 ou CS3 (CS = ***chip select).*** Dans la colonne activée où il n'y a pas de jonction diode diffusée sur l'une ou plusieurs des rangées A, B, C ou D, celles-ci comportent 1 bit de valeur « 1 » (niveau $\mathscr{H}$).

Il est bien évident que le coût de revient d'une ROM fabriquée en un seul exemplaire est trop élevé et que pour amortir le prix du masque et de sa diffusion, il est nécessaire d'avoir besoin d'une quantité importante de ROM du même modèle. Le prix de la plupart d'entre elles ne devient rentable qu'à partir d'un minimum de 100 à 150 pièces. Sinon, pour des quantités moindres on peut avoir recours à une ***PROM.***

Réalisation d'une ROM à diodes comportant 4 mots de 4 bits

Figure 73.



Le bus d'adresses comporte deux lignes A_0 et A_1 qui peuvent définir dans la mémoire les 4 adresses binaires suivantes :

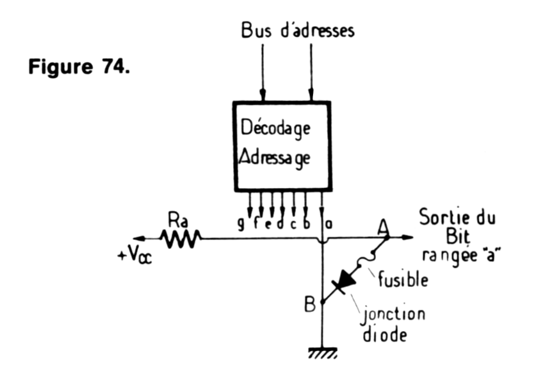
	A_0		A_1				
1re adresse	0	-	0	→	Colonne 1	→	1 0 1 0
2e adresse	0	-	1	→	Colonne 2	→	1 0 0 1
3e adresse	1	-	0	→	Colonne 3	→	0 1 1 0
4e adresse	1	-	1	→	Colonne 4	→	1 0 1 1
	adresses à 2 bits				Rangées	→	A B C D

Chacune de ces adresses correspond à l'une des colonnes : 1, 2, 3 ou 4. Les adresses indiquées sont décodées par un décodeur d'adresse qui active l'une des colonnes (ordre de lecture donné par le ***chip select (CS)*** L'exemple ci-dessus montre comment CS activé, l'adresse de la colonne 2 permet de lire dans les rangées A, B, C, D le mot de 4 bits à accès parallèle 1 0 0 1

Les ***PROM*** (Program ROM) sont des ***ROM programmables*** (sous-entendu par l'utilisateur). Elles sont achetées vierges par l'utilisateur. Leur technologie de fabrication est telle que celui-ci peut lui-même les inscrire selon son désir de programmation. Il existe plusieurs sortes de fabrication. On distingue en particulier les ***PROM à fusible*** et les ***PROM à destruction de jonction.*** Dans les 2 cas toutes les jonctions diodes entre colonnes et rangées sont diffusées par le fabricant.

Exemple de Prom à fusible

Figure 74.



Si l'on fait passer un fort courant entre rangée a et Colonne a c'est-à-dire entre les points A et B, le fusible claque ; cette coupure inscrit au point de croisement entre A et B le bit « 1 » (niveau $\mathcal{H}$).
Si l'on ne fait pas passer de courant le bit « 0 » (niveau $\mathcal{B}$) est inscrit en permanence.

Dans *les jonctions PROM à fusible* (voir figure 74) la diode est montée en série avec un fusible aluminium ou nickel-chrome.

Il suffit à l'utilisateur de faire passer un fort courant dans la colonne et la rangée qui détermine le point choisi pour faire fondre le fusible et couper le contact avec la diode diffusée. On peut alors y lire le bit « 1 » (niveau $\mathcal{H}$).

Dans les *PROM à destruction de jonction* (voir Fig. n° 75) on appliquera une tension inverse suffisante pour faire claquer la diode (2) créant ainsi à sa place un court-circuit du point correspondant de mémoire. On peut ainsi y lire un bit « 0 » (niveau $\mathcal{B}$).

Les PROM si elles reviennent moins cher à l'unité que les ROM, sont à l'inverse beaucoup plus coûteuses pour des produits en quantités de même modèle.

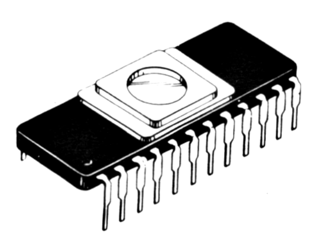
Exemple de PROM à destruction de jonction

Figure 75

Entre A et B, 2 jonctions tête-bêche maintiennent à la sortie sur la rangée a le bit « 1 » (niveau $\mathscr{H}$) du fait de la diode 2 montée en inverse.
Entre C et D, si l'on applique une tension inverse suffisante, on claque la diode 2 qui se comporte alors comme un court-circuit. A la sortie sur la rangée b on lit le bit « 0 » (niveau $\mathscr{B}$)

Figure 76

Exemple de AROM



On aperçoit sur le dessus du boîtier et au centre la fenêtre de quartz.

Il existe maintenant des mémoires PROM que l'on peut effacer, puis, ré-enregistrer. Elles s'appellent des ***REPROM.*** Ces mémoires sont généralement assez difficiles à effacer et à réenregistrer, aussi, elles ont été aujourd'hui pratiquement abandonnées au profit des ***EPROM*** et des ***AROM.***

EPROM (Erasable PROM), soit ***PROM effaçable*** et ***AROM*** (Altérable ROM), soit ***ROM altérable,*** dans le sens également de effaçable, sont toutes deux des mémoires de technique assez similaire.

Ces mémoires à circuits intégrés comportent sur leur boîtier une petite fenêtre d'ouverture en quartz afin d'être sensibles aux rayons ultra-violets ***(UV).***

Une exposition moyenne de 10 minutes dans des conditions de rayonnement bien spécifiées suffit à effacer toute programmation.

Les circuits sont alors prêts pour être électriquement reprogrammés

3. Etude des mémoires faisant partie de la famille MOTOROLA - M6800

1. *Mémoire RAM* - ***MCM6810A*** (voir Fig. 77)

C'est une ***RAM statique*** technologie MOS. Canal N - gate silicium, c'est une mémoire de ***128 mots de 8 bits.***

Elle est compatible avec tous les produits de la famille M6800 et de la Famille TTL (Alimentation simple de + 5 V).

Elle est organisée pour être utilisée directement, sans buffers, avec un système interface de lignes BUS.

Soit : BUS de données - D0 à D7 (mots de 8 bits, accès parallèle)
Bus d'adresses A0 à A6 (7 lignes soit 128 mots)
+ 4 « chips sélect » $\overline{CS1}$ - $\overline{CS2}$ - $\overline{CS4}$ et $\overline{CS5}$ qui activent le décodeur d'adresses.

BUS de contrôle : $\overline{CS3}$ (Validation adresse → Mémoire)
R/W (read - write)
(E) CSO - validation des données
Alimentation + 5 volts et masse.

— Le cycle de lecture mémoire est de 450 nanosecondes minimum.

— Le temps d'accès-adresse est de 450 nanosecondes maximum.

— Le cycle d'écriture mémoire est de 450 nanosecondes minimum.

— La durée d'écriture de l'information est au minimum de 190 ns.

— Le temps de maintien d'adresse est de 10 ns minimum.

2. *Mémoire ROM :* ***MCM6830A*** (voir Fig. 78)

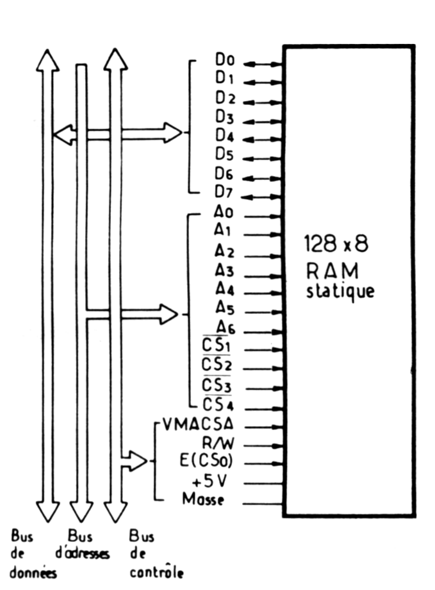
C'est une ***ROM statique*** technologie MOS - Canal N - gate silicium. C'est une mémoire de ***1024 mots de 8 bits.*** Elle est compatible, avec les produits de la famille du micro-ordinateur M6800 et aussi de la famille TTL. Elle est organisée pour être utilisée directement sans buffer avec un système interface de lignes BUS - soit :

— BUS de données : D0 à D7 (mots de 8 bits)

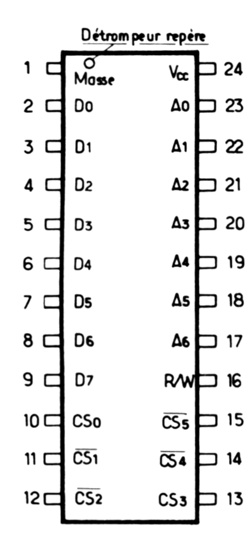
— BUS d'adresses : A0 à A9 (10 lignes soit 1024 mots : 2^{10}) + 3 chips « Select » de décodage d'adresses : CS1 - CS2 et CS3

— BUS de contrôle : E(CS0) - validation donnée + alimentation : +5 V et masse.

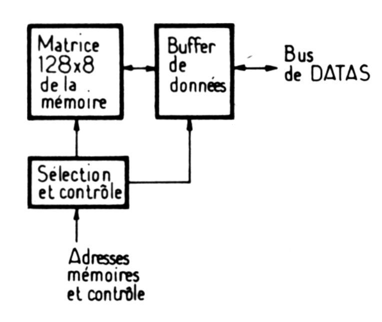
Figure 77 **Mémoire RAM - *MCM6810 A***



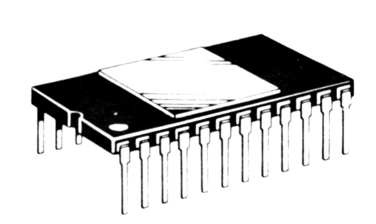
Liaisons de la mémoire avec bus d'interface microprocesseur



Ordonnancement du boîtier

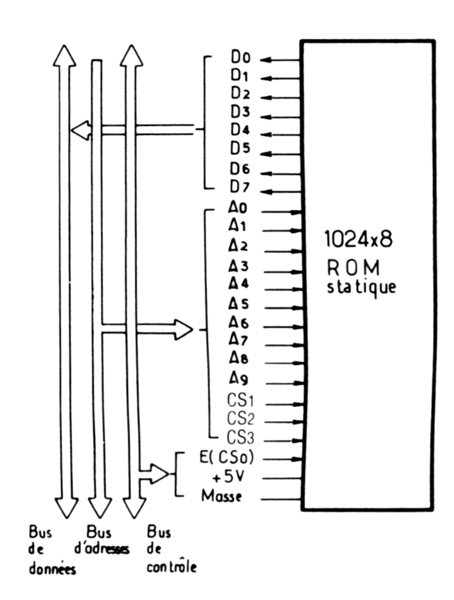


Bloc diagramme d'accès à la mémoire

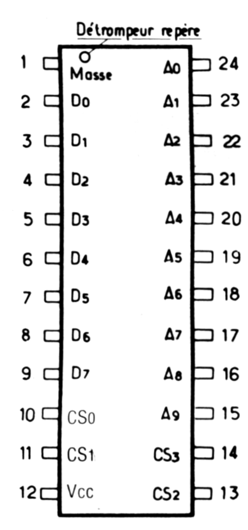


Boîtier DIL 24 connexions

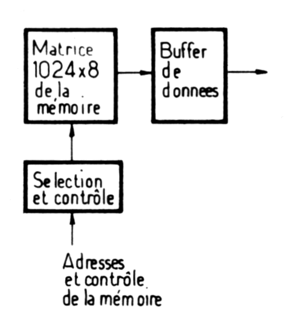
Figure 78 **Mémoire ROM - *MCM6830 A***

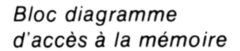


Liaison de la mémoire au bus d'interface du microprocesseur

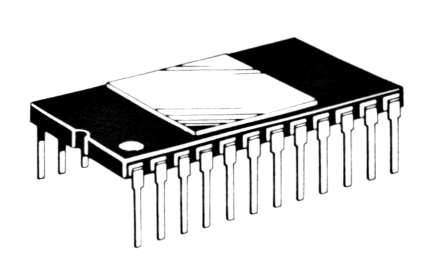


Ordonnancement du boîtier





Bloc diagramme d'accès à la mémoire



Boîtier DIL 24 connexions

Les niveaux d'activation d'entrée des chips « selects » et du contenu de la mémoire sont définis par le client. Les sorties des informations sont à 3 états (TSC).

Le cycle de lecture mémoire est de 500 ns minimum.

Le temps d'accès d'adresse est de 500 ns maximum.

A titre indicatif, pour terminer nous figurons ici la liste des mémoires de la famille M6800 dont l'utilisateur peut disposer.

MCM6810A - RAM statique 128 x 8.
MCM6830 et ***MCM6830A*** - ROM - 1024 x 8
MCM6832 - ROM - 2048 x 8
MCM68317 - ROM 2048 x 8
MCM6604 et ***MCM 6605A*** RAM dynamique 4096 x 1

Boîtier DIL 16 broches	Boîtier DIL 22 broches

CHAPITRE **11**

Les circuits et systèmes d'interface entre microprocesseur / mémoires unité centrale et périphériques

Les circuits d'interface dans un micro-ordinateur réalisent la liaison entre d'une part le microprocesseur et les mémoires de l'unité centrale et d'autre part les périphériques.

Les circuits d'interfaces constituent le système d'***I/O*** (input-output) ou le système ***d'entrées-sorties*** des informations du périphérique vers le micro-ordinateur et vice-versa.

Pourquoi un tel système ? Parce qu'il existe de par leur nature même des ***incompatibilités*** de fonctionnement entre un périphérique et son micro-ordinateur et que pour ***converser*** entre eux, il est nécessaire de monter en tampon un circuit d'interface capable d'adapter le fonctionnement de l'un au fonctionnement de l'autre. Ces ***adaptations*** doivent souvent s'effectuer selon 3 niveaux :

1er niveau

Une ***adaptation de temps.*** En effet un micro-ordinateur a des vitesses de travail de par la conception de son hardware purement électronique généralement supérieures à celles d'un périphérique dont l'appareillage comporte souvent bon nombre d'éléments de fonctionnement mécaniques ou électro-mécaniques.

2e niveau

Une ***adaptation de logique.*** La logique du périphérique peut être en effet différente de celle du micro-ordinateur.

3e niveau

Une ***adaptation de format de données.*** Le micro-ordinateur reçoit des mots suivant un ***accès parallèle,*** par contre le périphérique transmettra, la plupart du temps, eu égard à son système de support de programmation, toutes les données ***en série.***

Le système d'interface va permettre la ***translation série-parallèle*** des informations.

Voyons maintenant de quelle manière peut s'effectuer le système de commande d'un automatisme industriel.

Le fonctionnement de cet automatisme est vérifié, disons à l'aide de ***capteurs.*** Ces capteurs enregistrent les données de fonctionnement de l'automatisme (c'est-à-dire : séquences de déroulement d'une façon chronologique des opérations, mesures quantitatives et qualitatives de ces séquences). Ils vont ensuite les transmettre à un périphérique d'ordinateur (cette transmission peut s'opérer via un circuit ***convertisseur analogique - Digital).*** Le périphérique va alors interroger le micro-ordinateur dans le but de contrôler les données qu'il a reçues et les ***comparer*** au programme préalablement enregistré dans une mémoire de l'unité centrale. Cette vérification étant effectuée, si besoin est, une opération quantifiée de demande de modification va alors être transmise au périphérique qui agira en conséquences sur les circuits de commande de l'automatisme.

Au cours de ces opérations, il y a donc eu échange d'informations à l'aller :

1) De l'automatisme au périphérique
2) Du périphérique au circuit d'interface I/O
3) Du circuit d'interface I/O au Microprocesseur et à ses mémoires centrales.

Puis, en retour, un échange d'informations empruntant un parcours inverse. Il est donc intéressant de connaître maintenant comment peut s'opérer dans cette chaîne de transmission, l'échange des informations entre le micro-ordinateur et son périphérique.

L'interface I/O afin d'assurer cet échange doit comporter un ***registre d'état*** contrôlant le périphérique.

(Est-il, par exemple, en état à un instant précis, d'adresser une donnée au micro-ordinateur, ou bien est-il prêt à en recevoir une. N'est-il pas plutôt en train de rebobiner une bande magnétique de sa mémoire... ?)

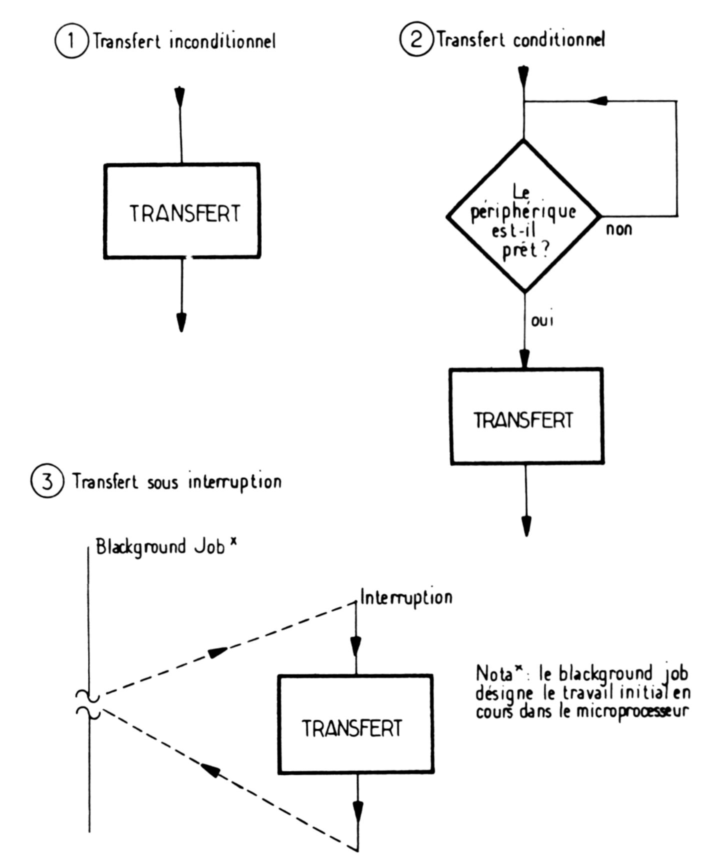
L'interface I/O devra également posséder un ***registre d'informations*** permettant l'échange des données.

Cet échange peut se réaliser, sous 4 conditions différentes, ce sont :

1. Le ***transfert inconditionnel***
2. Le ***transfert conditionnel***
3. Le ***transfert sous interruption***
4. Le ***transfert avec interruptions multiples***

Ces différents transferts sont illustrés dans la figure 79.

Figure 79 **Schémas de présentation des transferts d'I/O**



1. Le transfert inconditionnel

On ne teste pas le registre d'état car on transfère directement les données au périphérique. Il est alors nécessaire de connaître parfaitement bien ses temps de disponibilités, ce qui n'est pas toujours évident.

2. Le transfert conditionnel

On teste le registre d'état afin de savoir si le périphérique est disponible. Le transfert de données a lieu dès que la réponse est positive.

3. Le transfert sous interruption

C'est évidemment le moyen le plus sûr et le plus rapide. Ce problème a d'ailleurs déjà été traité au cours des chapitres 2 (voir Fig. n° 3) et 8 (paragraphe 7d) et aux explications des instructions ***IRQ*** et ***NMI*** du MC6800.

4. Le transfert avec interruptions multiples

Cela peut être le cas lorsque un assez grand nombre de périphériques sont reliés ensembles au même micro-ordinateur.

a) priorité de la demande. Si 2 périphériques ① et ② font simultanément une demande d'interruption, le fait de ***tester*** le registre d'état de ① d'abord, puis de ② ensuite, assure une ***priorité d'interruption*** à ①;

b) priorité de la routine de service - ① va donc effectuer : ***sa routine de service.*** Le bit ***i*** du microprocesseur est à « 1 » et effectue le ***masquage*** des demandes des autres périphériques.

Cependant, si la routine de service comporte une instruction remettant le bit « I » (Interrupt), du registre à « 0 », la routine de service, peut à son tour être interrompue par une demande du périphérique ② qui va alors posséder cette fois une priorité de service supérieure à celle de ① .

Enfin pour terminer ce problème de transfert de données, rappelons le cas particulier du travail en ***DMA*** (accès direct à la mémoire de l'unité centrale). Revoir chapitre 2, figure 4. Ce procédé exécute le transfert de données entre mémoires de l'unité centrale et périphérique.

Le microprocesseur pendant cette opération est masqué du fait de l'impédance infinie reportée à son entrée (***TSC*** = Three State Control).

Exemples de circuits d'interface

Nous allons aborder maintenant des exemples de circuits d'interface.

Tout d'abord le ***PIA*** possédant une liaison avec le périphérique assurée par un BUS datas formé de 8 lignes (mots de 8 bits à ***accès parallèle).***

Ensuite l'***ACIA*** possédant une liaison avec le périphérique assurée par une ligne d'informations à ***accès série.*** Ainsi dans ce second exemple, nous observons que le circuit d'interface adapte le format de données. Il permet une ***translation série-parallèle*** des informations afin que le périphérique puisse converser avec le ***MPU.***

1. Le PIA - (Peripheral Interface Adapter) est un circuit d'interface intégré (présentation en figure 80), faisant partie des produits de la famille M6800 de MOTOROLA. Il a le gros avantage de pouvoir mettre directement en communication le micro-ordinateur avec 2 périphériques extérieurs.

Le PIA - ***MC6821*** - est un circuit intégré MOS. Canal N gate silicium. Il est présenté en boîtier DIL à 40 broches de sorties.

Il comporte deux parties distinctes :

La 1re partie qui assure la liaison directe sans buffer externe avec les BUS du MPU et la 2e partie assurant la liaison directe sans buffer externe avec les 2 BUS de datas des 2 périphériques branchés.

1re partie - Le PIA comprend :

1. 8 lignes *bidirectionnelles* ***(D0 à D7)*** d'informations (mots de 8 bits à accès parallèle). Elles établissent la liaison avec le « BUS de datas » du MPU.

2. Les lignes du « BUS de contrôle » qui sont :

a) ***la ligne R/W*** (Read-Write). Le signal est généré par le MPU afin de contrôler la direction de transfert des données sur le « BUS de datas ». Un niveau logique « $\mathcal{B}$ » sur la ligne R/W du PIA valide l'entrée des buffers et la donnée se trouve transférée du MPU au PIA (MPU → PIA). Un niveau logique « $\mathcal{H}$ » sur la ligne R/W du PIA dispose celui-ci à un transfert de données vers le « BUS de DATA » du MPU (PIA → MPU).

b) ***La ligne « E » -*** ligne de Timing (temps d'exécution de transfert des données). Elle reçoit le signal d'horloge dérivé du MPU ***(clock*** Q_2***).***

c) ***La ligne*** $\overline{\textbf{RESET}}$. Le niveau « $\mathcal{B}$ » d'activation de la ligne est employé pour remettre tous les bits des registres du PIA à zéro. Elle sert à la ***« réinitialisation »*** (redémarrage) du PIA après une coupure d'alimentation.

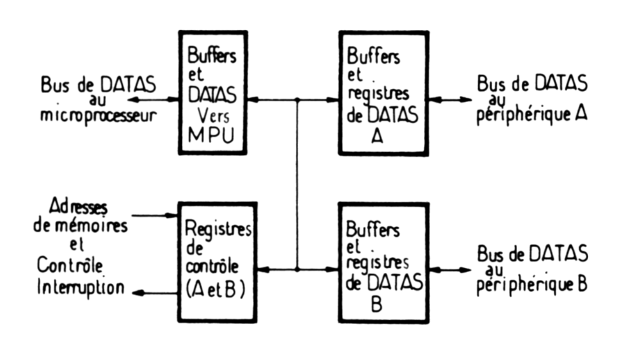
Figure 80 Le circuit d'interface - PIA - *MC6821*

Détrompeur repère

Broche	Signal	Signal	Broche
1	V_{ss}	CA_1	40
2	PA_0	CA_2	39
3	PA_1	$\overline{IRQA}$	38
4	PA_2	$\overline{IRQB}$	37
5	PA_3	RS_0	36
6	PA_4	RS_1	35
7	PA_5	$\overline{RESET}$	34
8	PA_6	D_0	33
9	PA_7	D_1	32
10	PB_0	D_2	31
11	PB_1	D_3	30
12	PB_2	D_4	29
13	PB_3	D_5	28
14	PB_4	D_6	27
15	PB_5	D_7	26
16	PB_6	E	25
17	PB_7	CS_1	24
18	CB_1	$\overline{CS_2}$	23
19	CB_2	CS_0	22
20	V_{cc}	$R/\overline{W}$	21

Détermination des connexions du PIA

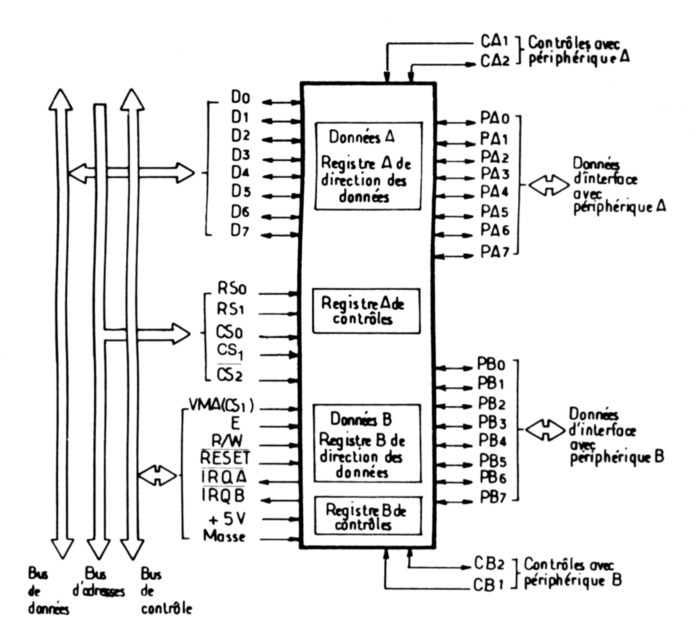
Boîtier DIL 40 connexions



Bloc diagramme du PIA

d) $\overline{\textbf{IRQA}}$ et $\overline{\textbf{IRQB}}$. Le niveau «$\mathcal{B}$» d'activation sert à effectuer une demande d'interruption du périphérique A ou B, soit directement au MPU ou soit à travers un circuit d'interruption prioritaire.

Le circuit d'interface - PIA - *MC6821*



e) ***CS0, CS1*** et $\overline{\textbf{CS2}}$**.** Ce sont les 3 ***chips select*** qui doivent être activés afin d'établir la liaison entre le « BUS de datas » et le PIA, les transferts s'effectuant toujours sous le contrôle des lignes (E) et (R/W).

f) ***RS0*** et ***RS1.*** Ce sont les signaux de contrôle chargés de sélectionner le registre (A ou B) dans lequel doit s'opérer l'écriture ou la lecture, des informations.

2[e] *partie.* Elle comprend :

La section A et la section B du PIA.

La section A du PIA est affectée au périphérique A.

La section B du PIA est affectée au périphérique B.

La section A relie le périphérique A au circuit d'interface grâce aux lignes ***PA0*** à ***PA7*** porteuses de mots de 8 bits à accès parallèle.

Chacune de ces lignes étant bilatérales, leur sens est programmé par *le registre A de direction de données.*

Si l'indicateur (flag) de ce registre est à « 1 », ces lignes se comportent comme des sorties (PIA → périphérique).

Si l'indicateur (flag) de ce registre est à « 0 », ces lignes se comportent comme des entrées (périphérique → PIA).

Toutes ces lignes admettent des niveaux logiques compatibles avec ceux de la famille des circuits TTL.

La section B relie le périphérique B au circuit d'interface grâce aux lignes de données ***PB0*** à ***PB7.*** Tout ce que nous venons de dire de la section A pour le périphérique A s'applique à la section B pour le périphérique B. Cependant on peut ajouter que les lignes ***PB0*** à ***PB7*** ont également la capacité d'admettre une logique à 3 états (TSC), ce qui leur permet dans le sens périphérique → PIA de posséder des ***entrées à haute impédance.***

Dans le sens PIA → périphérique, elles peuvent encore être employées comme ***source de courant*** de 1 mA sous 1,5 V et commander ainsi, si besoin est, directement la base d'un transistor de commutation.

Les ***lignes d'entrées CA1 et CB1 :*** ces lignes d'entrées adressent une demande d'interruption de la part des périphériques A et B ***aux registres de contrôle*** correspondants A et B.

La ***ligne de contrôle CA2*** du périphérique A. Cette ligne bilatérale (CA2) sert à définir une interruption d'entrée ou bien encore sert à contrôler la sortie vers le périphérique.

La fonction de cette ligne est programmée par le *registre A de contrôle.*

La ***ligne de contrôle CB2*** du périphérique B est une ligne bilatérale qui est le pendant de la ligne CA2. Ce que l'on dit de CA2 s'applique à CB2 envers le périphérique B.

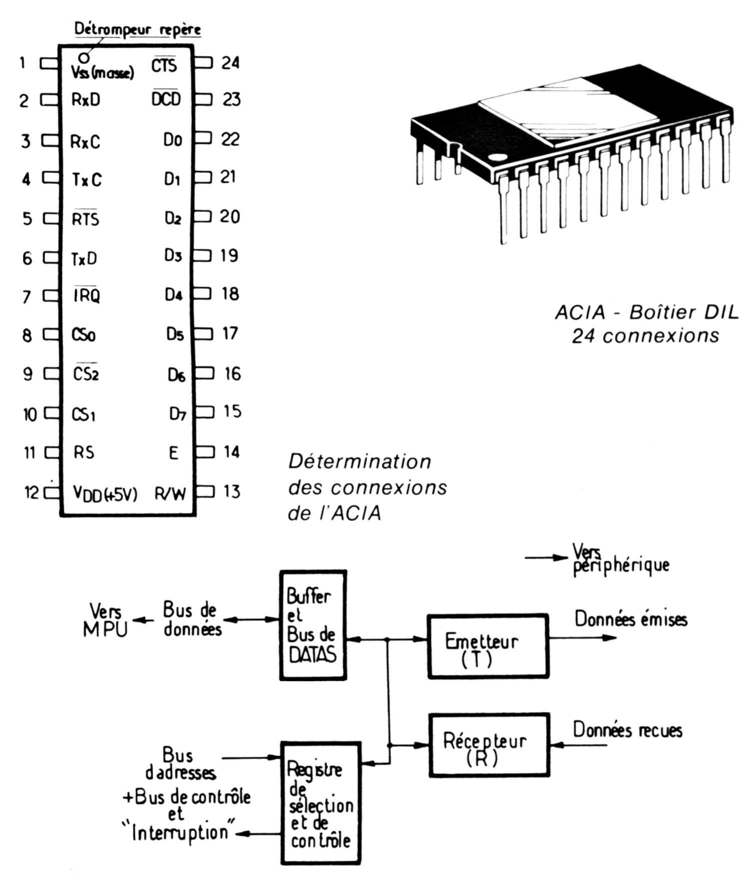
Cependant, il faut ajouter que dans le sens (périphérique → PIA) son entrée est ***à haute impédance*** et dans le sens contraire, cette ligne peut servir de ***source de courant*** de 1 mA, sous 1,5 V et peut directement commander, si besoin est, la base d'un transistor de commutation.

Les 2 lignes (CA2) et (CB2) présentent des caractéristiques de compatibilité avec celles de la famille TTL.

2. L'ACIA. (Asynchronous Communications Interface Adapter) de MOTOROLA ***MC6850*** est un autre type de circuit d'interface différent du PIA (voir Fig. 81). C'est un circuit intégré MOS. Canal N. gate silicium. Il est encapsulé dans un boîtier DIL comprenant 24 broches de sorties.

Il transmet les informations au MPU - MC6800 à l'aide de mots de 8 bits à « accès parallèle » grâce à un « BUS de datas » comportant les lignes ***D0 à D7.***

Figure 81 **Le circuit d'interface ACIA - *MC6850***



ACIA - Boîtier DIL 24 connexions

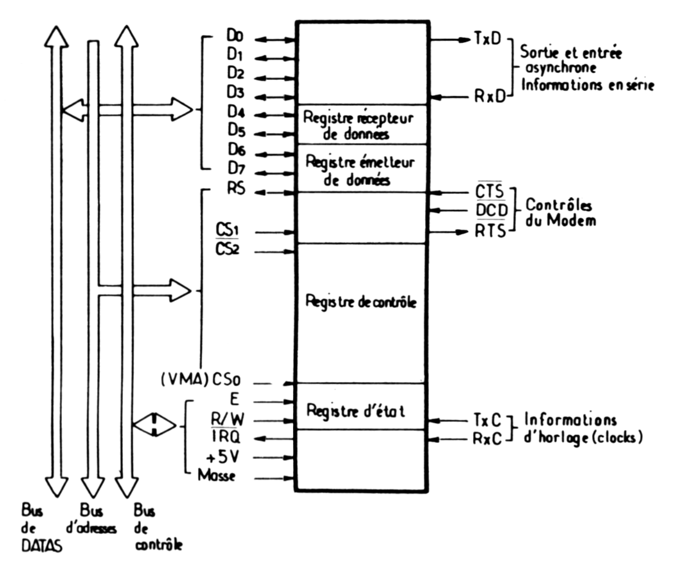
Détermination des connexions de l'ACIA

Bloc diagramme de l'ACIA

Les informations reçues du périphérique ne sont pas dans le ***format accès-parallèle*** mais dans le ***format à accès série.*** Le cadencement de ces données est ***asynchrone.***

Afin que le MPU reçoive correctement les informations du périphérique il y a donc ***translation « série parallèle »*** du format des données à l'inté-

Figure 81 (suite) Le circuit d'interface - ACIA - *MC6850*



rieur de l'ACIA. A l'inverse, lorsque le MPU répond au périphérique, il y a ***translation « parallèle-série »*** des données à l'intérieur de l'ACIA.

Entre le périphérique et le MPU, l'ACIA peut communiquer via un ***MODEM.***

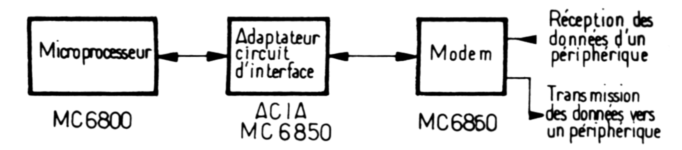
Un modem est un matériel spécifique qui permet de porter à très grande vitesse et également sur des distances de plusieurs kilomètres, sur le réseau P et T des données en provenance d'un périphérique (le MPU les reçoit) ou en provenance d'un MPU (le MPU les transmet au périphérique). La liaison s'effectue suivant la représentation synoptique de la figure n° 82.

Le modem lorsqu'il reçoit des informations en provenance d'un périphérique démodule le courant porteur afin de transmettre à l'ACIA les informations démodulées du périphérique, et, lorsqu'il transmet des informations du MPU vers l'ACIA, il les module afin de les porter sur le réseau téléphonique.

L'ACIA monté en tampon ***translate les formats des données*** reçues du périphérique (translation série parallèle) ou adressées au périphérique (translation parallèle-série).

Figure 82

Synoptique d'une liaison : microprocesseur → Périphérique



MODEM signifie « ***Mod***ulateur - ***Dém***odulateur »

Nous allons maintenant décrire rapidement l'ACIA.

Dans l'ACIA, on distingue deux registres :

Un ***registre récepteur de données*** recevant sur la ligne (***R x D***) les informations à « accès-série » en provenance du périphérique et un ***registre de données*** transmettant sur la ligne (***T x D***) les informations à « accès-série » vers le périphérique.

Ces 2 registres sont contrôlés par un ***registre d'état*** et un ***registre de contrôle*** eux-mêmes activés extérieurement à l'aide de lignes de contrôle. Ces ***lignes de contrôle*** reçoivent les signaux idoines, soit en provenance du MPU ou soit y sont adressées.

Des signaux spéciaux sont émis pour le contrôle du modem extérieur à l'ACIA ($\overline{\textbf{CTS}}$ - $\overline{\textbf{DCD}}$ - $\overline{\textbf{RTS}}$) ainsi que les signaux de synchronisation d'horloge qui sont dirigés dans l'ACIA (***T x C*** et ***R x C***). Pour plus amples détails, il est bon de se reporter à la bibliothèque MOTOROLA, ouvrage (M6800 - Microcomputer - System Design Data).

Le contrôleur de priorité d'interruption (PIC).

Ce circuit introduit ***une notion de priorité dans les réponses aux signaux d'entrée reçus par le système.*** Dès que le MPU détecte une interruption et y répond, le PIC choisi dans la ROM l'adresse correcte de départ du sous-programme de service correspondant. Le PIC classique de la famille MOTOROLA - M6800 est le circuit bipolaire **MC6828.**

Le contrôleur de disque souple.

Ce circuit assure les fonctions complexes d'interface entre le MPU et une unité de disque souple. Aidé d'un multiplexeur extérieur, un même coupleur peut contrôler plusieurs unités de disques souples.

Le contrôleur classique de la famille MOTOROLA - M6800 de disque souple est le circuit LSI à 40 broches **MC6843.**

Le coupleur d'accès direct à la mémoire (DMAC) travaille avec un coupleur de périphérique et un générateur d'horloge MPU afin ***de faciliter l'accès direct du circuit périphérique à la mémoire du système sans qu'il y ait intervention de la part du microprocesseur.*** Le coupleur (DMAC) de la famille MOTOROLA M6800 est le **MC6844.**
Le Timer Programmable (MC6840) génère des signaux carrés et sert à la commande et au contrôle des Intervalles de temps.

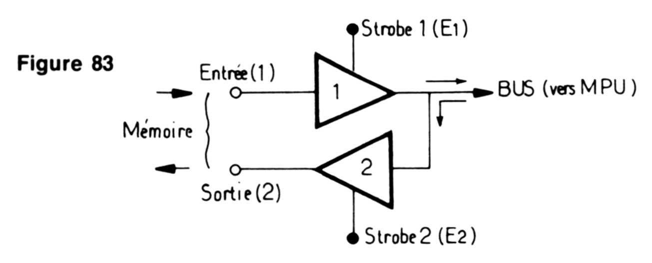
Les amplificateurs de BUS : « émetteur-récepteur »

Dans certains cas, la liaison entre un BUS de lignes bilatérales (exemple : bus de données) et le microprocesseur doit se faire via un circuit d'interface communément appelé : ***transceiver,*** soit ***émetteur-récepteur.*** Ce circuit est notamment nécessaire lorsque le bus de données relie au microprocesseur un système très chargé de circuits mémoires et de circuits d'interfaces I/O. Dans ce cas le BUS en question est souvent appelé avec le transceiver un ***BUS-expanseur*** (bus extender).

Il s'agit en réalité de 2 amplificateurs montés « tête-bêche » qui amplifient en courant les signaux en provenance du MPU vers les mémoires (signal d'écriture « émetteur ») et des mémoires vers le MPU (signal de lecture « récepteur ». Le schéma de principe en est donné à la figure n° 83.

Synoptique d'un circuit d'interface « *Transceiver* »

Figure 83



Exemple de réalisation
Le Quad (quatre) Transceiver
(Emetteur-Récepteur) de MOTOROLA ***MC8T26***

On voit comment le microprocesseur opère pour « lire », venant de la mémoire, « une donnée » à l'entrée (1) via le « BUS de datas ».

Le ***strobe « 1 »*** ou ***enable*** E_1 autorise l'amplificateur ▷1 à s'ouvrir à la donnée et l'amplifie en courant alors que le ***strobe 2*** ou ***enable 2*** interdit l'accès de toute donnée. Pour cela, l'amplificateur ◁2 voit son entrée placée en régime de haute impédance (***TSC***). A l'inverse, pour écrire une donnée, venant du MPU dans la mémoire, le ***strobe « 2 »*** ou ***enable « 2 »*** autorise l'amplificateur ◁2 à s'ouvrir au passage de la donnée tout en l'amplifiant en courant, alors que le ***strobe « 1 » (E_1)*** interdit l'accès à toute donnée, l'amplificateur ▷1 voyant son entrée placée en régime de haute-impédance ***(TSC).***

Vers des périphériques économiques... !

Dans ce chapitre, il a été question des circuits d'interface permettant le dialogue entre un micro-ordinateur et les périphériques.

Rappelons que si, aujourd'hui, on peut se procurer un kit câblé de micro-ordinateur assurant un programme minimum et comportant outre le microprocesseur, une RAM, une ROM et un circuit I/O le tout avec un circuit d'horloge à moins de 3 000 francs, un ***télétype*** ou tout autre périphérique du genre coûte hélas ! au moins 5 fois plus cher.

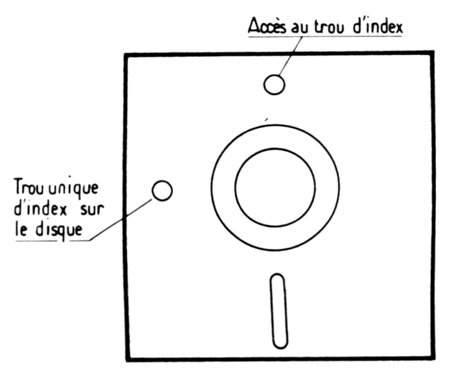
Cependant, aujourd'hui les récentes ***unités de disquettes (floppy-disk)*** de faible volume et de prix plus abordables permettent enfin aux périphériques de réaliser un ensemble de choix pour les applications mettant en œuvre les micro-ordinateurs. La disquette (voir Fig. 84) est un disque souple de 18 cm de diamètre recouvert sur une face d'***oxyde magnétique*** et protégée par une pochette en carton. Les autres caractéristiques de ces disquettes peuvent varier dans des proportions assez faibles d'un modèle à un autre. A titre d'informations générales nous en donnons les caractéristiques approximatives suivantes :

— nombre de pistes : 77
— densité d'inscription : 48 pistes/par pouce (2,54 cm)
— capacité d'insertion en mégabits : 3,1
— vitesse de transfert en kilo bits par seconde : 250
— vitesse de rotation des disquettes : 360 tours/minute

On envisage également avec l'utilisation des micro-ordinateurs l'emploi pour les périphériques de petites ***cassettes standard magnétiques.*** Certaines sont déjà dans le commerce. Leur capacité est de l'ordre de 300 000 caractères par bande avec une vitesse de déroulement de 120 caractères à la seconde.

La ***transmission de caractères*** (chiffres et lettres majuscules et minuscules plus certains signes) se fait par l'intermédiaire d'un ***codage interface.***

Figure 84 **Dessin de présentation de « Disquette »**



Citons à titre d'exemple, quelques types de disquettes actuellement disponibles en France :

Calcomp 140	*Memorex 651*
Control Data 9400	*Memorex 652*
DRI74	*Potter 4740*
IBM 3740	*Sagem D53*
Logabax 45	

Il existe deux codes principaux :

Le code ASCII (American Standard Code for Information Interchange). Code d'échange standard américain d'informations et le ***Code EBCDIC*** (Extended Binary Coded Decimal Interchange Code). Le code d'échange étendu au système BCD.

On monte maintenant des microprocesseurs avec leurs circuits d'interface directement dans des périphériques réalisant ainsi ce que l'on appelle des ***terminaux « lourds »*** ou bien encore des ***terminaux « intelligents ».***

Les moniteurs T.V. M68MDM1 et M68MDM9

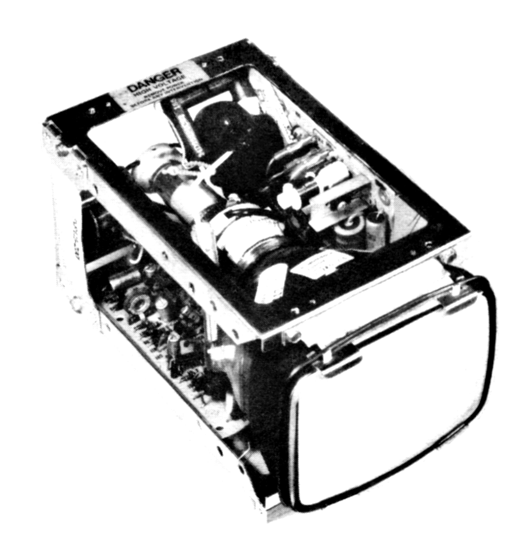
1. Généralités

Ces moniteurs Vidéo-fréquences de TV sont entièrement transistorisés et servent à afficher, sur leur écran, des caractères alphabétiques et numériques. Les modèles M68MDM fonctionnent soit avec une entrée de signal « ***vidéo-composite*** » ou bien encore avec des entrées séparées pour :

— Un signal ***vidéo-fréquences***
— Un signal de ***synchronisation-trame***
— Un signal de ***synchronisation lignes***

Dans ce dernier cas les niveaux de signaux de synchronisation sont compatibles avec les niveaux de la logique ***TTL*** (voir figure 85).

Les tubes image employés ***C.R.T.*** (cathode ray tube) sont soit des « 5 pouces » (le ***M68MDM1***) soit des 9 pouces (le ***M68MDM9***). Cette numérotation en pouces indique la longueur de la diagonale des écrans. Rappelons qu'un pouce est égal à 2,54 cm.



Le moniteur TV M68 MD M1

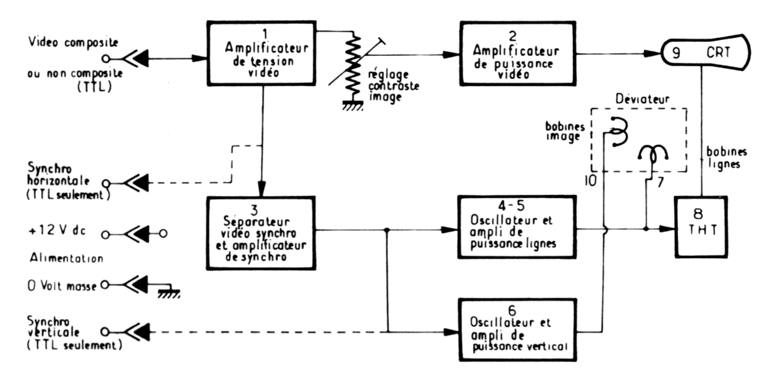


Figure 85 (Schéma Synoptique des moniteurs TV)

Les tubes image sont « ***auto-protégés*** » c'est-à-dire « ***anti-implosion*** » et du type à déviation magnétique.

Le châssis du moniteur ne comporte pas l'alimentation.

Une alimentation continue de 12 volts/650 mA extérieure au châssis est donc nécessaire.

Une sortie est utilisée en option dans le cas d'une commande à distance de la brillance de l'écran du moniteur.

2. Caractéristiques électriques essentielles — (Voir tableau de la figure 86).

3. Réalisation mécanique du châssis des moniteurs T.V.

Le câblage comporte essentiellement deux circuits imprimés :

Une carte des « Signaux efficaces vidéo et des oscillateurs driver de balayage horizontal et vertical » S_1 et une carte des « Circuits de puissance de déflexion du tube image D_2.

Un connecteur de 10 broches situé sur le côté de la carte D_2 sert au branchement des entrées et des sorties du moniteur.

4. Principe de fonctionnement du moniteur (voir figure 85)

Il comprend :

a) ***Un amplificateur du signal image vidéo-fréquences*** formé de 4 transistors (1 et 2). On a la possibilité de régler le contraste image (1) ainsi que

Tableau figure 86

Spécifications électriques des moniteurs TV

Tube image CRT	**Angle de déviation « Diagonale 55° » Phosphore standard : P4**	
Alimentation DC	**12 volts — 650 mA**	
Signaux d'entrées	Cas du signal Vidéo-Composite	Entre 0,5 et 2,5 volts crête/crête synchro négative Ze = 75Ω bouclé ou 12 kΩ non rebouclé
	Cas des synchros Horizontal et Vertical niveaux TTL	2,5 à 5 V crête Synchro positive Ze = 75 à 250 Ω sur la connection vidéo Ze > 2 kΩ pour les synchros Horizontal et vertical
Définition (finesse image)	650 points au centre de l'image 500 points sur les bords de l'écran	
Bande passante Vidéo	10 Hz à 12 MHz à — 3 dB	
Linéarité	< 2 % mesurée suivant les normes EIA	
THT	9,5 kvolts avec un courant de faisceau du tube image de 50 μA	
Temps de retour lignes	Maximum 11 μs	
Fréquence de balayage	en horizontal : 15 750 Hz ± 500 Hz en vertical : 50 Hz à 60 Hz	
Environnement	Température de fonctionnement : 0 °C à 50 °C Température de stockage : — 40 °C à + 65 °C	

la polarisation d'entrée de l'étage de puissance Vidéo final (2) ce qui assure le réglage correct du courant de repos de cet étage. Des protections sont assurées au niveau de la limitation du courant de faisceau du tube cathodique et de la remontée des « ***Arcings*** » (claquages) inter-électrodes du tube image vers les étages vidéo du signal image.

b) ***Un circuit de séparation vidéo-synchro du signal vidéo-composite*** (3) qui fournit séparément les signaux de synchronisation des balayages lignes et image du tube cathodique. Ce circuit, dans le cas de vidéo non-composite, peut également servir d'amplificateur des signaux de synchronisation du balayage lignes lorsqu'ils sont externes à la carte de câblage et compatibles TTL (voir tableau figure 8, niveaux de compatibilité).

c) ***Un circuit oscillateur verrouillé à la fréquence du balayage lignes*** (4) grâce à la tension de détection d'erreur de phase entre les tops de synchronisation et les impulsions du retour de balayage lignes.

d) ***Un circuit de puissance de balayage lignes*** (5) chargé d'alimenter les bobines de déflexion lignes (7), les électrodes d'accélération et de concentration du CRT (9) et de générer la THT (8). A noter encore la possibilité dans ce circuit de règler la largeur du balayage lignes.

e) ***Un oscillateur vertical et son amplificateur de puissance*** (6) chargé d'alimenter, les bobines de déflexion verticale (10). A noter dans ce circuit la possibilité de régler correctement la linéarité et l'amplitude du balayage image vertical.

Le générateur de caractères vidéo - Motorola - MC6847

Le générateur de caractères Motorola ***(VDG = Vidéo display generator) MC6847*** est un circuit intégré servant ***d'interface*** entre un micro-système comportant d'une part les microprocesseurs MC68000 et MC6800 associés à leurs diverses mémoires compatibles et d'autre part un moniteur vidéo-fréquences monté dans une console de visualisation périphérique. Le ***VDG*** lit dans une mémoire du microsystème une donnée qu'il traduira en signal vidéo-composite. Ce signal injecté à l'entrée d'un moniteur vidéo ***inscrira sur l'écran de son tube image des caractères alphabétiques et numériques*** (voir figure 87).

Le MC6847 est un circuit intégré ***N-MOS*** encapsulé dans un boîtier de ***40 broches de sorties*** disposées suivant une configuration « ***Dual in line*** » (double rangée en ligne) semblable à celle du boîtier photographié en figure 80.

Chaque page de texte émis peut comporter ***16 lignes de 32 caractères alphanumériques par ligne.***

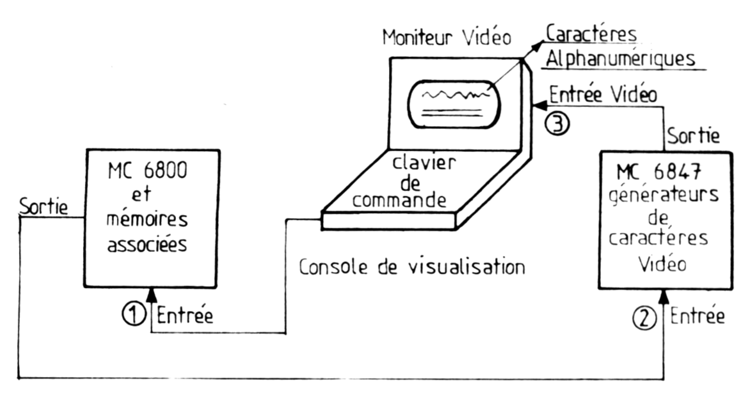


Figure 87

Une ROM d'instructions interne au générateur de caractères à masque programmable, est disponible sur commande spéciale.

Le MC6847 comprend 2 modèles. L'un avec balayage lignes ***non entrelacé*** et l'autre le ***MC6847Y*** avec le balayage lignes ***entrelacé.***

Ce circuit intégré génère également des signaux de différence de couleurs ***R-Y et B-Y.***

Il est compatible avec le circuit intégré modulateur Motorola ***MC1372.*** L'ensemble MC6847 et MC1372 peut alors être branché à la prise antenne soit d'un TV noir et blanc, soit d'un TV couleurs afin d'afficher sur son écran des signaux ***« alphanumériques »*** programmés à partir d'un micro-système M6800. C'est à partir d'un tel ensemble que l'on peut inclure les ***« Jeux vidéo ».***

Le modulateur MC1372 reçoit :

1) Le signal de luminance vidéo

2) Les signaux de différence de couleurs R-Y et B-Y à la sortie du MC6847.

D'autre part dans le MC1372 figurent

1) Un circuit générant ***une porteuse HF*** située dans la bande I de TV (Canaux 3 ou 4) et un circuit dans lequel cette porteuse est modulée par le signal de luminance.

2) Un circuit générant ***une sous-porteuse chrominance*** pilotée à l'aide d'un quartz extérieur au MC1372 et un circuit dans lequel cette sous-porteuse est modulée par les signaux de différence de couleurs R-Y et B-Y.

*3) **Un circuit mélangeur*** mixant la porteuse HF et la sous-porteuse chroma toutes deux modulées.

La sortie du circuit mélangeur du MC1372 peut alors être branchée à ***la prise antenne d'un TV classique couleurs*** ou ***d'un TV noir et blanc*** réglé en bande I de TV (Canaux 3 ou 4).

Le montage d'ensemble est représenté en figure 88.

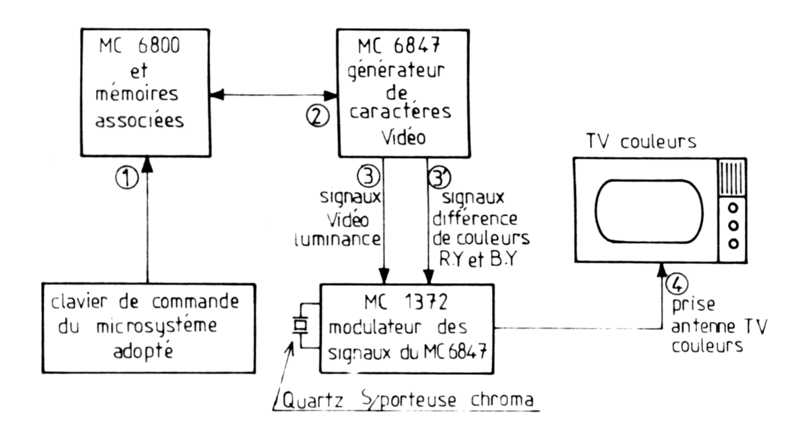


Figure 88

CHAPITRE **12**

La programmation Comment bâtir un micro-ordinateur en vue d'une utilisation donnée

Il faut bien se pénétrer de cette vérité première qui doit être acceptée tel un postulat avec lequel on ne peut absolument pas transiger : le micro-ordinateur, c'est une machine ***bête et disciplinée.***

Cette expression montre bien ***le caractère mental*** de la machine et nous fait mieux comprendre comment on devra s'en servir si l'on désire qu'elle nous renseigne efficacement.

En effet si l'on veut que le micro-ordinateur :

— Stocke les informations
— Réalise des calculs

et prenne des décisions, le programmeur devra effectuer un programme comportant une suite d'instructions.

Chacune de ces instructions a pour but de prévoir et dicter pas à pas tous les détails de fonctionnement et de contrôle ***(tests)*** auxquels la machine doit se conformer.

La programmation est donc très importante et nécessite, une bonne compréhension d'organisation du micro-ordinateur et une bonne connaissance de son langage d'expression.

Elle nécessite encore de la part du programmeur beaucoup d'ordre et de méthode dans l'élaboration de son programme.

Afin qu'il en soit ainsi, il lui est recommandé de le concevoir en le composant à partir du ***cycle chronologique*** suivant :

1.1. Rechercher la définition du problème à résoudre

Cette recherche doit être aussi minutieuse que possible. Ainsi pour gérer un contrôle d'automatisme, il faut bien définir les différentes sé-

quences à contrôler, c'est-à-dire : leur ordre d'exécution, leur temps de durée, leur qualité... etc.

1.2. Choisir la méthode la mieux adaptée pour la machine en vue de parvenir à la résolution du problème posé

Les méthodes à mettre en œuvre pour résoudre un problème sont parfois assez nombreuses. Il convient donc de rechercher celle qui cadre le mieux avec les qualités du microprocesseur employé. Une représentation graphique souvent appelée ***flow-chart*** peut en fixer clairement les idées. Nous en examinerons ci-après les modalités d'exécution.

1.3. Appliquer la méthode adaptée à l'aide du choix judicieux d'un ordre logique de séquences d'instructions à programmer

C'est ce que l'on désigne sous le vocable de ***codage du problème.*** Cette opération conduit à stocker dans la mémoire centrale du micro-ordinateur le programme d'exécution du problème devant être résolu par la machine. A ce stade, on pourrait penser que la programmation est ainsi achevée et que l'homme ayant ainsi supplé à la *« bêtise de la machine »*, celle-ci *« disciplinée »* possède ainsi les moyens de résoudre dans défaillir le problème posé.

Et bien... ! cela n'est pas encore certain et en dernier ressort, il faut :

1.4. Vérifier, pour se rendre compte que l'homme a été *« assez intelligent »* dans la compréhension de l'effet de chaque instruction déterminée à l'intérieur de l'ensemble du programme réalisé. Cette vérification peut s'obtenir à partir d'un ***programme de vérification*** appelé par les Américains : ***debug program.***

Maintenant que nous nous sommes fixés une idée générale sur la conception, par le programmeur d'un cycle théorique chronologique d'établissement d'un programme, nous allons étudier comment s'y prendre afin de le réaliser d'une façon pratique ; pour cela nous allons successivement entreprendre :

2.1. De vous donner un aperçu sur la représentation graphique des ***symboles d'un organigramme*** (flow-chart)

2.2. De vous décrire la manière dont on peut réaliser ***l'introduction d'un programme par étapes graduelles*** dans un micro-ordinateur venant d'être installé et mis en service.

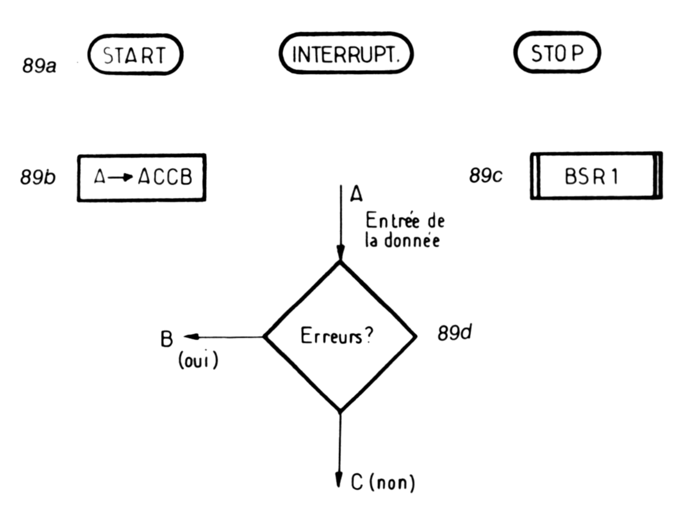
2.3. Nanti de ces connaissances, nous résoudrons un problème pratique simple.

2.4. Enfin pour terminer, nous mentionnerons quelques-uns des ***outils de travail*** mis en œuvre, à titre d'exemple, par MOTOROLA pour se servir d'une manière efficace de son microprocesseur MC6800.

2.1. Les symboles pratiques d'établissement d'un organigramme (flow-chart)

Ceux-ci sont représentés dans la figure 89.

Représentation des différents symboles d'un organigramme



89a. Cet ovale représente suivant l'inscription écrite dedans :

Un ***début,*** une ***interruption*** ou une ***fin de programme.***

89b. Ce rectangle représente, suivant l'inscription écrite dedans, ***la tâche à développer en cours de programme*** (ici par exemple la donnée A doit être transférée dans l'accumulateur B, on dit encore : chargez A dans l'accumulateur B.

89c - Représentation dans le programme en cours du ***développement d'un sous-programme extérieur*** (ici par exemple : branchement à la subroutine 1).

89d - Le losange est le symbole de représentation d'un ***test.*** Ainsi on voit ici à titre d'exemple la figuration à l'intérieur du losange d'un ***test***

d'erreurs. La donnée entre en A. Elle est soumise au test d'erreurs. S'il y a effectivement une erreur, la donnée sort en B où elle ira vers une séquence de ***correction d'erreurs*** avant d'entrer à nouveau en A et ressortir enfin en C.

S'il n'y a pas d'erreur, la donnée sort directement en C et le programme continue à se dérouler normalement.

2.2. L'introduction d'un programme par étapes graduelles

On peut distinguer deux grandes catégories de programmes différents : les ***programmes utilitaires*** qui préparent le programme d'application pour lequel l'ordinateur est requis et les ***programmes d'applications.***

a) Les programmes utilitaires : ils comportent une séquence d'***initialisation*** comprenant elle-même deux étapes distinctes :

1re étape. Sur le panneau frontal d'un micro-ordinateur, toute une série de clés (interrupteurs) déterminent en les actionnant les bits d'un mot, ainsi que des fonctions directement en langage machine (binaire naturel). En agissant ainsi sur ces clés, on démarre la première étape qui consiste à ouvrir le dialogue avec le périphérique. C'est le rôle du ***bootstrap*** exécutant le ***programme d'amorçage.***

2e étape. Ce bootstrap est maintenant suivi d'un ***programme de chargement*** ou ***chargeur absolu.*** Cette fois ce programme est introduit via le clavier du télé-imprimeur du périphérique. Il prépare le programme d'applications en fixant les adresses réelles en mémoire, en vérifiant les formats d'adresses et de données, en détectant les erreurs ***(programme d'aide)*** etc. Afin d'éviter la servitude d'exécution par l'opérateur de toute l'étape d'initialisation, le bootstrap, comme le chargeur absolu peuvent être stockés dans une mémoire (ROM) de l'unité centrale du micro-ordinateur. Ce travail étant accompli, on peut alors implanter les ***programmes de service et d'applications.***

b) Les programmes de service et d'applications

Dans ces programmes d'applications on peut distinguer :

L'***édition du programme source*** et l'***édition du programme objet.***

Programme source d'applications : on effectue ce programme en langage mnémonique. Il peut être finement ajusté à l'aide de programmes de service appelés encore ***programmes de mise en forme*** parmi lesquels on peut mentionner : L'***éditeur de liens*** et l'***éditeur de textes.***

Les différentes étapes d'introduction d'un programme

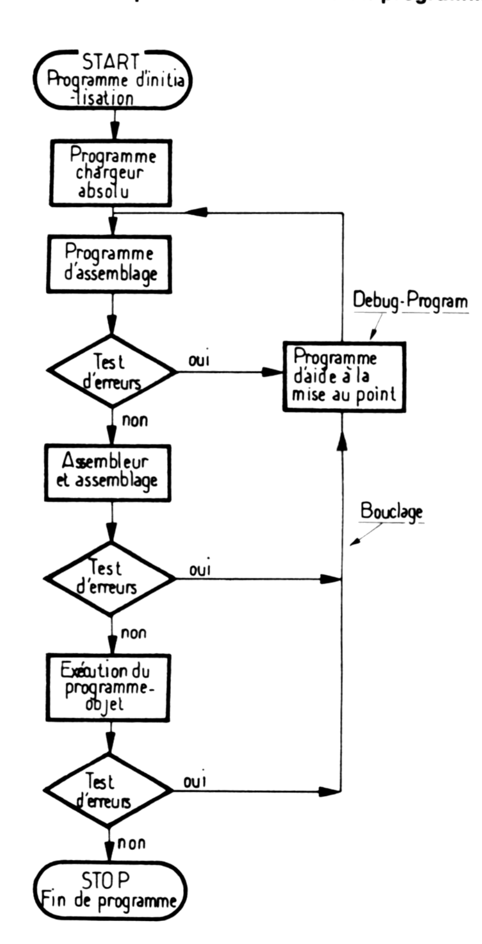


Figure 90

L'éditeur de liens (Chargeur - Editeur de liens) a pour but de réunir dans l'ordre choisi tous les modules programmes séparés nécessaires à l'élaboration d'un programme complet d'application.

L'***éditeur de textes*** corrige et édite les programmes source. Il exécute par exemple un ordre d'annulation d'un caractère erroné sur une bande,

ou bien encore le programme étant en mémoire, il peut en supprimer une instruction, la déplacer, la remplacer par une autre... etc. Il sert en somme à éditer le programme source définitif. Lorsque l'éditeur de texte fait partie intégrante du système du micro-ordinateur, il est appelé : ***éditeur résident.***

Programme objet d'application. A la demande du programmeur, l'assembleur va coder le langage source en langage machine (binaire naturel). Viendra alors l'étape d'exécution du programme objet. Cette exécution étant achevée après d'éventuels tests d'erreurs et de corrections, s'il en est besoin, le programme doit être ***stoppé*** dans le micro-ordinateur. La figure 90 résume ce paragraphe 2-2 concernant l'introduction d'un programme par étapes graduelles.

2.3. Résolution d'un problème pratique simple :

En reprenant notre étude théorique du début de chapitre pour l'appliquer d'une manière concrète, nous allons successivement :

a) Rechercher la définition d'un problème simple à résoudre.

b) Définir l'adoption d'une méthode conduisant à l'établissement d'un organigramme de base ***(Flow-Chart).***

c) Appliquer enfin cette méthode en définissant le ***codage du problème.***

Exemple de problème à résoudre : nous élaborerons notre solution à l'aide d'un jeu d'instructions imaginé, vraisemblable.

a) Définition : soit à résoudre l'opération : $\boxed{4 - 3 + 2}$. Cela signifie qu'il faut soustraire 3 de 4 et ajouter ensuite 2 à ce premier résultat partiel. Soit enfin charger l'information résultante dans la mémoire de l'unité centrale.

b) Choix de la méthode à adopter : nous allons choisir un mode d'***adressage direct*** (en supposant que les données : 4, 3 et 2 sont adressées dans une mémoire RAM du système du micro).

Afin d'effectuer la soustraction (4 − 3) nous allons nous servir de la méthode dite de ***soustraction par complémentation*** que nous avons exposée au cours du chapitre 5 concernant le calcul binaire au paragraphe soustraction.

Nous allons successivement chercher :

Le ***complément à 1*** du nombre $\mathbf{3_{10}}$ écrit en binaire, puis : ***incrémenter*** (+ 1) ce complément à 1 afin d'obtenir le ***complément à 2*** du nombre $\mathbf{3_{10}}$.

Il nous suffira alors d'ajouter à ce complément à 2, les nombres $\mathbf{4_{10}}$ et $\mathbf{2_{10}}$ pour obtenir le résultat demandé et le ***charger*** dans la mémoire de l'unité centrale.

On peut donc maintenant dresser le ***flow-chart*** de la figure n° 78 et qui sera, ici, explicite à l'aide d'un texte d'accompagnement devant le rendre plus clair à notre compréhension.

Un tel exemple d'organigramme établi à l'aide de cases les unes au-dessus des autres constitue un ***schéma de programmation en lignes droites*** (straight-line programming).

c) Codage du problème. Soit à ***coder les séquences d'instructions*** (voir tableau de la figure n° 92) afin de résoudre l'équation : $4 - 3 + 2$ selon un mode d'adressage direct.

Figure 91

Organigramme du programme devant résoudre l'opération : *4 — 3 + 2*

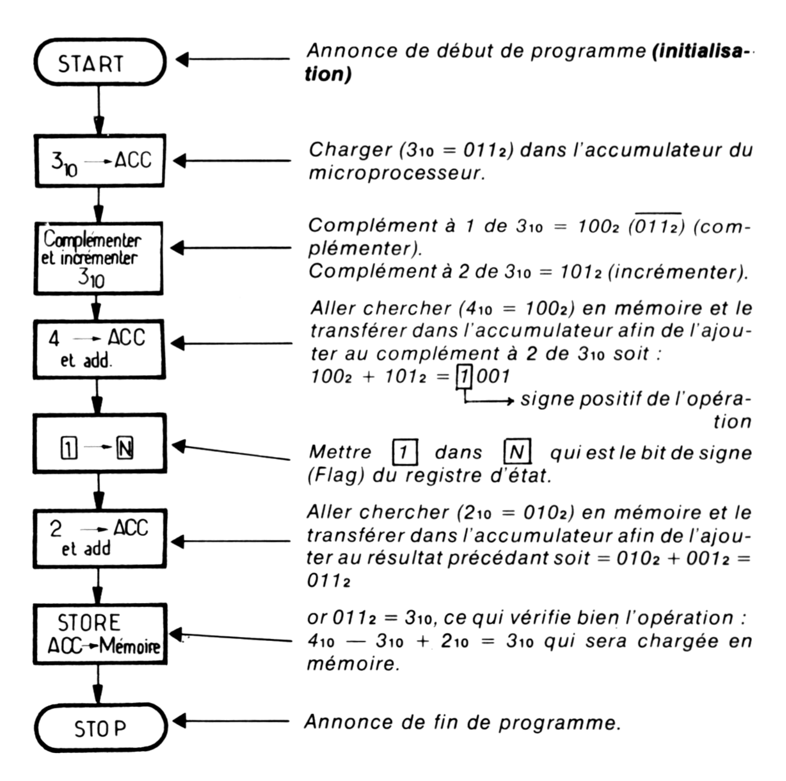


Figure 92

Tableau des séquences d'instructions devant résoudre le programme d'opérations : *4 — 3 + 2*, selon un mode d'adressage direct

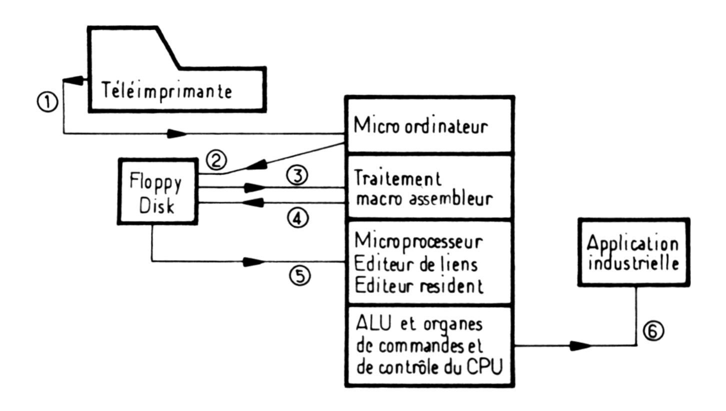
MODE D'ADRESSAGE DIRECT

Adresse	Instruction mnémonique	Traduction de l'instruction	Opération arithmétique ou booléenne
150	***CLR***	Remise à zéro (clear)	ACC = 0 (ACC = Accumulateur)
151	***ADD***	Additionner (159)	3 → ACC
152	***COM***	Complémenter	ACC = $\overline{3}_{10}$
153	***INC***	Incrémenter	ACC = $\overline{3}_{10} + 1$
154	***ADD***	Additionner (158)	4_{10} → ACC (+)
155	***ADD***	Additionner (160)	2_{10} → ACC (+)
156	***STR***	Charger Résultat (161)	ACC → Mémoire
157	***STP***	Arrêt	
158		4_{10}	
159		3_{10}	
160		2_{10}	
161		Résultat $(4 - 3 + 2)_{10}$	
162			

↑	↑	↑	↑
Adressage mémoire	Instructions imaginées (vraisemblables)	Explication en clair de la séquence effectuée	La flèche (→) indique le transfert

2.4. Les outils de travail

Soit l'installation de base figurée dans le schéma simplifié de la **figure 93** ci-dessous :



① → Le clavier de la ***téléimprimante*** va servir au programmeur pour composer le programme source (en langage mnémonique).

② → Le Programme source sera alors mis en mémoire dans le ***floppy-disk*** (utilisé en extension à la mémoire centrale du micro-ordinateur).

③ → Le ***macro-assembleur*** du micro-ordinateur va alors composer le programme objet à partir du programme source enregistré sur floppy-disk.

④ → On peut ensuite mettre en mémoire ce programme objet (langage binaire pur) sur le floppy-disk.

⑤ → On charge enfin ce programme objet dans le microprocesseur ainsi que d'autres ***modules objet*** à l'aide du ***chargeur-éditeur de liens*** et avec encore le ***programme de mise au point*** de l'éditeur résident afin d'obtenir le programme objet définitif corrigé, assurant la commande et le contrôle en ⑥ → de l'application industrielle recherchée.

Après ces considérations générales sur les outils de travail, il est bon d'entrer dans le domaine de la pratique. C'est ce que nous ferons en vous décrivant un des outils de travail très intéressant mis à la disposition du concepteur. Il s'agit de l'***exorciser*** de MOTOROLA s/c. Cet exorciser mis en service, conjointement avec un floppy-disk (***exordisk*** de MOTOROLA)

ou bien encore avec un lecteur de télétype (***exortape*** de MOTOROLA) va pouvoir concevoir et tester un prototype construit avec le MPU Kit (M6800).

L'exorciser de base ***(M68SDT)*** comporte :

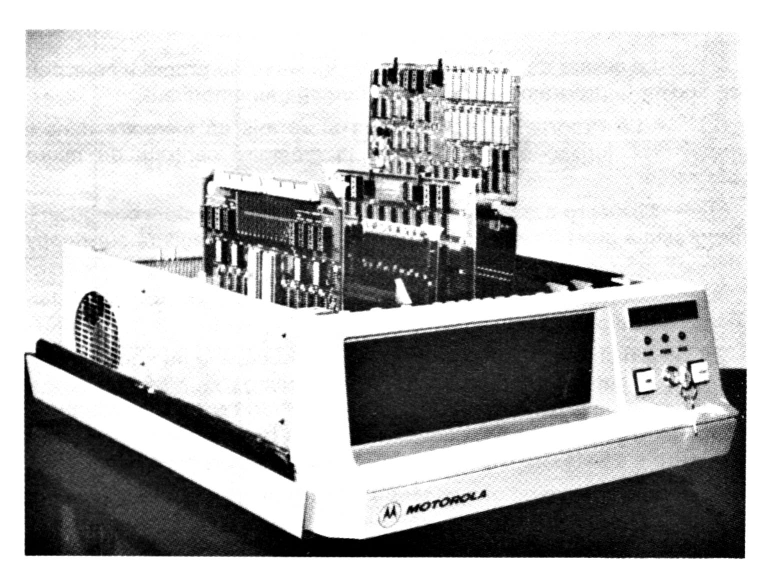
a) Un module MPU dans lequel est incorporé le circuit intégré MC6800 ainsi que son système d'horloge.

Ce module est appelé : ***MEX6800***

b) Un module de vérification de systèmes **(Debug module)** donnant à l'exorciser la possibilité d'évaluer et de procéder à la mise au point du programme de l'utilisateur.

Ce module contient entre autres 2 mémoires RAM (MCM6810) et 3 mémoires ROM (MCM6830). Dans ces dernières est inscrit un programme firmware appelé ***EXBUG*** qui est un ***programme de simulation.***

La simulation permet de calculer, outre les conséquences de chaque instruction sur le micro-ordinateur, le temps réel total d'exécution du programme désiré. Le ***simulateur*** est l'appareil définissant une simulation de programme.



EXORciser avec quelques cartes

c) Un module d'interface dont la possibilité de vitesse d'entrées et de sorties des informations peut être comprise par bonds successifs entre 110 et 9600 ***baud*** (soit entre 11 et 96 caractères par seconde).

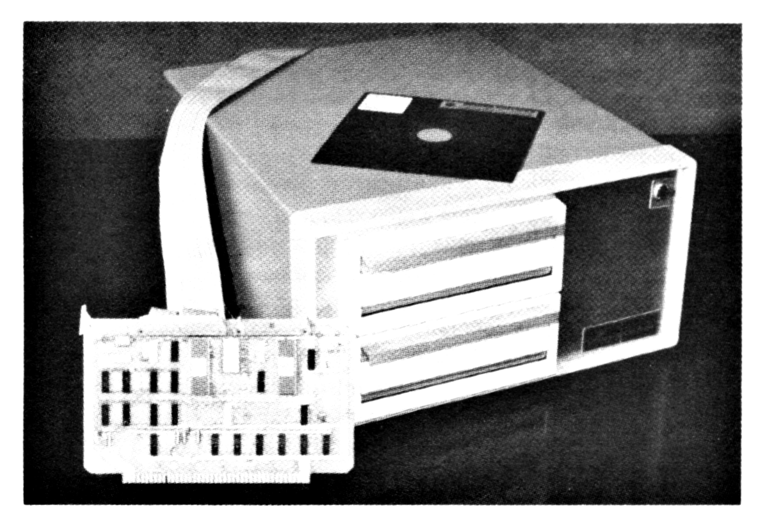
Ce module situé dans l'exorciser permet donc l'adaptation facile avec un terminal donné.

d) Une alimentation comportant 3 sources : + 5 Vdc, + 12 Vdc, et, — 12 Vdc.

e) Un châssis de câblage et de logement des modules. Il existe en réalité deux possibilités : un modèle de table ***(M68SDT-T)*** et un modèle en rack ***(M68SDT-R).*** On peut noter que l'alimentation et le module d'interface sont câblés directement sur le châssis, alors que le module MPU et le « debug module » sont des cartes de circuits imprimés enfichables dans des logements du châssis.

Ce modèle de base peut par extensions optionnelles loger jusqu'à 14 modules enfichables, ce qui détermine une très grande souplesse d'emploi.

On peut, en effet, avec l'exorciser, grâce à cette conception, définir des



Floppy Disk (disque souple) avec carte d'interface pour EXORciser

systèmes très variés de configurations devant faire face à tous les besoins des utilisateurs.

Les principaux types de modules optionnels à inclure dans l'exorciser sont :

— le ***MEX6812-1*** (Module de RAM statique 2 K bits)
— le ***MEX6815-1*** (Module mémoire RAM dynamique 8 K bits)
— le ***MEX6821*** (Module d'entrées-sorties1/0)
— le ***MEX681C*** (C'est un câble d'interconnexions entre le MEX6820 et un périphérique),

— Le ***MEX68WW*** - C'est le système universel d'enclenchement mécanique et électrique des modules précités dans l'exorciser et permettant au client de simuler sa propre conception des circuits.
— Le ***MEX68XT (module d'extension).*** C'est une carte imprimée de connexions ordonnées qui permet de relier extérieurement à l'exorciser d'autres configurations créant ainsi de nouvelles extensions au système par rapport à celles déjà disponibles à l'intérieur même de l'exorciser.

Les modules optionnels permettent à l'ingénieur de développement de ***HARD*** de réaliser des économies appréciables de temps, car il peut au moyen de changements de cartes et de simples commutations sur celles-ci représenter son système. Ce « meccano » est ainsi, par rapport à un câblage conventionnel de prototype d'étude, l'occasion d'un réel gain de temps appréciable.

Enfin l'exorciser peut également en option posséder un ***M6800 software resident.*** Ce software comporte un ***editeur resident*** et un ***assembleur resident.***

Rappelons ici que l'editeur sert à écrire le programme source corrigé définitif et l'assembleur à le transposer en langage machine.

L'option « software resident » M6800 fournit ainsi au programmeur la possibilité de développer entièrement son propre software.

Utilisé conjointement avec son logiciel ***« firmware exbug »***, l'exorciser permet à l'ingénieur de « Hard », comme au programmeur de ***« soft »*** de vérifier et de corriger la configuration tant dans le domaine des circuits à utiliser que dans celui du programme de son système.

On réalise ainsi avec l'exorciser une mise au point spécifique complète et définitive du système adapté aux besoins réels du client.

L'exorciser M6800 de MOTOROLA est donc en conclusion, un excellent moyen économique de développement des applications de la clientèle avec le microprocesseur MC6800.

Conclusion

C'est ainsi que nous terminons le chapitre 12 concernant la programmation et la réalisation définitive d'un système.

Ce chapitre clôt ainsi ce livre d'initiation à la micro-informatique. A chaque nouvelle édition ce livre a été remis soigneusement à jour.

L'auteur est conscient d'avoir fait éditer un livre de débutant dans cette technique de la micro-informatique. Son ambition aura été de vous intéresser à sa découverte qui sans doute va vous permettre, du moins l'auteur l'espère, de mieux suivre les progrès de développement de cette nouvelle génération de l'électronique dans tous les domaines déjà existants : l'informatique, les télécommunications, le radar, les transmissions HF, les automatismes industriels, la télévision, la radiophonie... etc.

L'auteur a souhaité, tout au long de l'écriture de son texte, que ce livre crée en vous la curiosité et l'attrait nécessaires à l'intention d'étudier dans le but de devenir par la suite, un véritable spécialiste dans ce nouveau domaine prometteur d'activités.

S'il a ainsi pu vous inspirer et vous convaincre, il pense avoir rempli avec succès son contrat ; celui d'engager, pour sa très infime et sa très modeste part, la jeune génération montante des techniciens vers une nouvelle source communautaire de richesse intellectuelle et scientifique, génératrice de compréhension mutuelle par-delà les frontières géographiques et linguistiques de la société humaine universelle.

L'auteur

Bibliographie

Ce livre a essentiellement pris ses sources d'informations dans la bibliothèque MOTOROLA S/C Products Inc.

Les principaux ouvrages consultés ont été :

— M6800 - microprocesseur programming manual
— Microcomputer system - reference handbook
— Du computer au microprocesseur de **Hervé Tireford** (computer lab, Genève de MOTOROLA S/C)
— M6800 - Microcomputer - system design data
— Exorciser data sheets (november 1975)
— Bulletin MPU - M6800 de juin 1976
— M6800 - systems reference and data sheets.
— Composants MOTOROLA pour micro-ordinateurs.

Attention. A noter dans le texte de ce livre que les mots : ***Exorciser*** - ***Exbug*** - ***Exordisk*** et ***Exortape*** font l'objet de marques déposées par MOTOROLA S/C.

Merci à tous mes collègues de MOTOROLA qui d'une manière ou d'une autre ont contribué par leur autorisation, leurs renseignements et leurs conseils à la réussite de ce cours d'initiation sur les « MICROPROCESSEURS ».

Achevé d'imprimer sur les presses de l'Imprimerie Marcel Bon 70 Vesoul
Dépôt légal : juin 1984 - N° éditeur : 427 - N° imprimeur : 2785

MICRO SYSTEMES



INITIATION A LA MICRO-INFORMATIQUE : LE MICROPROCESSEUR

Cet ouvrage intéressera aussi bien le micro-informaticien que l'électronicien. Il apportera au premier un complément utile à l'étude de l'informatique par une explication détaillée des fonctions, des langages et de la programmation du microprocesseur. Au second, il donnera de nombreuses indications sur la technologie et l'évolution de ce composant aux innombrables applications.

Principaux chapitres :

- Cerveau humain et ordinateur
- Langages d'ordinateurs
- Calcul binaire et codages
- Fonctions logiques
- Evolution technique
- Divers types de mémoires
- Circuits et systèmes d'interface
- Programmation
- Technologie et organisation des microprocesseurs

ISBN 2-85535-086-7

TARIF CODE D

14

INITIATION A LA MICRO-INFORMATIQUE
LE MICROPROCESSEUR

P. MELUSSON

★



AMSTRAD
CPC

MÉMOIRE ÉCRITE

ACME

https://acpc.me/